UNIT 1 INTRODUCTION TO COMPUTER GRAPHICS

Structures **Page Nos.**

1.0 Introduction 5
1.1 Objectives 6
1.2 What is Computer Graphics? 6
1.3 Applications 7
1.3.1 Presentation Graphics 8
1.3.2 Painting and Drawing 10
1.3.3 Photo Editing 13
1.3.4 Scientific Visualisation 14
1.3.5 Image Processing 15
1.3.6 Education, Training, Entertainment and CAD 17
1.3.7 Simulations 21
1.3.8 Animation and Games 23
1.4 Graphics Hardware 25
1.4.1 Input and Output Devices 25
1.4.2 Display Devices 31
1.5 Summary 40
1.6 Solutions/Answers 41

1.0 INTRODUCTION

Early man used drawings to communicate even before he learnt to talk, write, or count. Incidentally, these ancient hieroglyphics (picture-writings) communicate as well today, as they must have done thousands of years ago, this fully supports the saying that "A picture is worth a thousand words" and in the era of computers we can add on to it or we may as well revise the saying to "*A computer is worth a million pictures!" ;* so, you can estimate the power of a computer as a communication system.

Now, with the advances in computer hardware and software, graphics has come a full circle and, more and more people are teaching and learning, communicating and sharing their ideas through the medium of graphics. By **graphics**, we mean any sketch, drawing, special artwork or other material that pictorially depict an object or a process or otherwise conveys information, as a supplement to or instead of written descriptions, and the utilisation of computers to accomplish such tasks leads to a new discipline of computer graphics. Traditionally, graphics has referred to engineering drawings of buildings, bridges, machine parts etc. and scientific drawings such as x-y curves, network and process flowcharts. In recent decades, graphics has ventured into industrial design, advertising and other artistic endeavours. During the last few years, even newspapers and periodicals aimed at the common man have begun to utilise graphics to present quantitative news such as selection results and production statistics. Computer graphics can do all this and more. In fact, the power and easy availability of computer graphics have increased the use of pictures to replace and augment words to describe, educate, or inform a wide variety of audiences, on a wide variety of subjects.

In this unit, we shall concentrate on the graphic capabilities and potential of the digital computer plus we will discuss the meaning of the term graphics and its types, in addition to which, we will also discuss the hardware used for practical application of graphics in different streams of life. The software section however, will be discussed in Block 4 of this course.

1.1 OBJECTIVES

After completing this unit, you should be able to:

- describe computer graphics, its features and characteristics;
- discuss applications of computer graphics in various fields, and
- describe various types of hardware, required to work with graphic systems.

1.2 WHAT IS COMPUTER GRAPHICS?

The meaning of the term Graphics, is Graphical Tricks. Every image or picture is in fact a graph and when different mathematical tricks are used to manipulate some change in its properties like shape, size, motion etc., through the help of computers then, the representation is nothing but computer graphics, so we can say that "*Computer Graphics (CG) is the field of visual computing, where one utilises computers both to generate visual images synthetically and to integrate or alter visual and spatial information sampled from the real world*." **Or** "*Computer Graphics is the pictorial representation manipulation of data by a computer*" **Or** "*Computer Graphics refers to any sketch, drawing, special artwork or other material generated with the help of computer to pictorially depict an object or a process or otherwise convey information, as a supplement to or instead of written descriptions*". Computer Graphics is a complex and diversified field. A Picture is a fundamental cohesive concept in Computer Graphics. Each picture consists of points called pixels (Picture- element). If we consider a complex picture, then complex database for pixels are considered, hence, complex algorithm are required to access them. These complex database contain data organised in various data structures such as ring structures,
B-tree etc.

In our earlier courses CS-60, we had learned mathematical tricks to do jugglery with the graphs that resulted from different functions, now let us learn how to juggle with graphics by using computers. There are many algorithms, which can be materialised to produce graphical effects on the screen through several graphical tools based on different languages that are available in the market.

Computer graphics can be broadly divided into the following classes:

- Business Graphics or the broader category of Presentation Graphics, which refers to graphics, such as bar-charts (also called histograms), pie-charts, pictograms (i.e., scaled symbols), x-y charts, etc. used to present quantitative information to inform and convince the audience.
- Scientific Graphics, such as x-y plots, curve-fitting, contour plots, system or program flowcharts etc.
- Scaled Drawings, such as architectural representations, drawings of buildings, bridges, and machines.
- Cartoons and artwork, including advertisements.
- Graphics User Interfaces (GUIs) which are the images that appear on almost all computer screens these days, designed to help the user utilise the software without having to refer to manuals or read a lot of text on the monitor.

We will discuss the various classes of computer graphics mentioned above in the following sections of this unit.

The most familiar and useful class of computer graphics involves movies and video games. Movies generally need graphics that are indistinguishable from physical reality, whereas video games need graphics that can be generated quickly enough to be perceived as smooth motion. These two needs are incompatible, but they define two-ways of communications between users and computations. In video games, the

subject matter of computations is generally characters chasing and shooting at each other. A more familiar use of computer graphics exists for interacting with scientific computations apart from movies and games. This familiarisation of the use of computer graphics has influenced our life, through simulations, virtual reality, animation, we can extend the scope of education, entertainment, analysis etc. So, in global terms Computer graphics can be categorised in two ways:

Interactive Computer Graphics which is interactively used by users e.g., games. We will discuss details about this type of graphic systems in Block 4.

Passive Computer Graphic which has no option for users to interact or use computer graphics e.g., movies. We will discuss details about this type of graphic systems in Block 4.

1.3 APPLICATIONS

Research in computer graphics covers a broad range of application including both photorealistic and non-photorealistic image synthesis, image-based modeling and rendering and other multi-resolution methods, curve and surface design, range scanning, surface reconstruction and modeling, motion capture, motion editing, physics-based modeling, animation, interactive 3D user interfaces, image editing and colour reproduction. Work is going on in various fields but computer vision is a hot topic, where research tackles the general problem of estimating properties of an object or scene through the processing of images, both 2D photographs and 3D range maps. Within this broad scope, we investigate efficient ways to model, capture, manipulate, retrieve, and visualise real-world objects and environments.

Once you get into computer graphics, you'll hear about all kinds of applications that do all kinds of things. This section will discuss not only the applications but also the software suitable for that type of application, so it is necessary to give you an understanding of what various applications do. While working on a project, you may need images, brochures, a newsletter, a PowerPoint presentation, poster, DVD etc. Thus, the question arises what software do I need to get my job done. The section will help to straighten all of that out in your head. Hopefully, if you know what does what, you won't waist money duplicating purchases, and when other designers or co-workers are talking shop, you'll know what is going on.

Graphic design applications are basically broken down on the basis of a few considerations. The first two considerations are, "Is your project for print, or web". When I say web, what I really mean is monitor based publishing. This means that you are going to see your work on a computer screen, and television, or a big screen projector. So, as you read through this section, whenever we say "web based", we mean monitor based. Beyond print and web, here are the various categories that we can think of that various applications would fit into; Image Manipulation; Vector Graphics; Page Layout; Web sight development; Presentation Software; Video Editing; DVD Production; Animation and Interactivity etc. If you are creating, or learning to create graphic design, computer art, or maybe "Digital Media" is the term that we should use, then it's a good thing to understand the function of each application. There are many applications in the market and most of them are expensive. A few of the various application areas that are influenced by Computer graphics are:

- Presentation Graphics
- Painting and Drawing
- Photo Editing
- Scientific Visualisation
- Image Processing

- Education, Training, Entertainment and CAD
- Simulations
- Animation and Games

Let us discuss these fields one by one.

1.3.1 Presentation Graphics

The moment you are going to represent yourself or your company or product or research paper etc. simply standing and speaking is quite ineffective. Now, in such a situation where no one stands with you, your ultimate companions are the slides which have some information either in the form of text, charts, graphs etc., which make your presentation effective. If you think more deeply, these are nothing but the ways some curves (text/graph/charts) which are used in some sequence and to create such things, graphics is the ultimate option, this application of graphics is known as presentation graphics, which can be done very effectively through computers now-a-days. There are some softwares which helps you present you and your concerned effectively. Such application softwares are known as **Presentation Graphics softwares – which is a software that shows information in the form of a slide show** (A slideshow is a display of a series of chosen images, which is done for artistic or instructional purposes. Slideshows are conducted by a presenter using an apparatus which could be a computer or a projector).

Three major functions of presentation graphics are:

- an editor that allows text to be inserted and formatted,
- a method for inserting and manipulating graphic images, and
- a slide-show system to display the content.

The program that helps users to create presentations such as visual aids, handouts, and overhead slides to process artwork, graphics, and text and produce a series of 'slides'– which help speakers get their message across are presenation graphics softwares.

Example programs include some softwares like Apple's Keynote, Openoffice's (Star Office-by Sun microsystems) Impress, Microsoft Powerpoint and (for multimedia presentations, incorporating moving pictures, and sounds) Macromedia Director. Custom graphics can also be created in other programs such as Adobe Photoshop or Adobe Illustrator and then imported. With the growth of video and digital photography, many programs that handle these types of media also include presentation functions for displaying them in a similar "slide show" format.

Similar to programming extensions for an Operating system or web browser, "add ons" or plugins for presentation programs can be used to enhance their capabilities. For example, it would be useful to export a PowerPoint presentation as a Flash animation or PDF document. This would make delivery through removable media or sharing over the Internet easier. Since PDF files are designed to be shared regardless of platform and most web browsers already have the plugin to view Flash files, these formats would allow presentations to be more widely accessible.

We may say that Presentation graphics is more than just power point presentation because it includes any type of slide presentation, bar chart, pie chart, graphs and multimedia presentation. The key advantage of this software is that it help you show abstracts of representation of work.

Note: There are some softwares like canvas that improves the presentation created through powerpoint or keynote software. Although these software packages contain a lot of handy features, they lack many vector and image creation capabilities, therefore, creating a need for a graphic/illustration program. Scientists, engineers, and other technically-oriented professionals often call

upon Canvas and its host of vector, image editing, and text features to create the exciting visual components for their presentation projects.

General questions that strike many graphic designers, students, and engineers rushing to import their illustrations and images into presentations are:

- What resolution should be used?
- Which file format is best?
- How do I keep the file size down?

Let us discuss in brief the suitability of the technique (Vector or Bitmap), and the file format appropriate to the creation of a better presentation.

Resolution

Graphic illustrations are used in presentations to help convey an idea or express a mood, two kinds of illustration graphics are:

1) Vector, and
2) Bitmap.

You may wonder which one of these is a better format when exporting to some software PowerPoint or Keynote or impress. The truth is that there are different situations that call for different methods, but here are some things to look out for. For instance, vectors are objects that are defined by anchor points and paths, while bitmapped graphics are digital images composed of pixels. **The advantage** of using vector graphics is that they are size independent, meaning that they could be resized with no loss in quality. Bitmapped graphics, on the other hand, provide a richer depth of colour but are size dependent and appear at the stated 72 dpi size**.**

File format

Say, we want an image of a Fly. The wings are partially transparent and to represent that in our presentation what be problematic if proper file format is not there. This choice of file format is hidden in the software that you may be using. Two cases for the same situation are discussed below:

- **The right file format that will allow us to create a transparent background in Keynote presentation.** Even though Keynote could import all common file formats such as GIF, JPG, and BMP, there is one format that will work particularly well which is .PSD. Using .PSD (Photoshop format) we are able to easily place a transparent image, even partially transparent sections of the image, such as the wings of the fly, as well as retain their properties.

- **The right file format that will allow us to create a transparent background in PowerPoint.** Even though PowerPoint could import all common file formats such as GIF, JPG, and BMP, there are two particular file formats that will work exceptionally well: TIFF and PNG. Using TIFF (Tagged-Image File Format) or PNG (Portable Network Graphic), we could easily remove the unwanted background quickly and easily in PowerPoint, a feature not available to the other mentioned file formats.

Note: TIFF or PNG: TIFF has been around longer than PNG, which was originally designed to replace GIF on the Web. PowerPoint works well with both these files when creating transparent backgrounds but generally PNG creates smaller file sizes with no loss of quality.

1.3.2 Painting and Drawing

When we talk about graphics, we mean pictures, and pictures can be either illustrations or photographs. If you want to get graphics into a Web page or multimedia presentation, you either have to create them in some kind of graphics application by drawing or painting them right there in the application, or bringing them into the application via a digital camera or scanner, and then editing and saving them in a form suitable to your medium.

Many software applications offer a variety of features for creating and editing pictures on the computer. Even multimedia authoring and word processing programs include some simple features for drawing on the computer.

So, painting and drawing application in computer graphics allows the user to pick and edit any object at any time. The basic difference is as follows:

Drawing in a software application means using tools that create "objects," such as squares, circles, lines or text, which the program treats as discrete units. If you draw a square in PowerPoint, for example, you can click anywhere on the square and move it around or resize it. It's an object, just like typing the letter "e" in a word processor. i.e., a drawing program allows a user to position standard shape (also called symbols, templates, or objects) which can be edited by translation, rotations and scaling operations on these shapes.

Painting functions, on the other hand, don't create objects. If you look at a computer screen, you'll see that it's made up of millions of tiny dots called pixels. You'll see the same thing in a simpler form if you look at the colour comics in the Sunday newspaper — lots of dots of different colour ink that form a picture. Unlike a drawing function, a paint function changes the colour of individual pixels based on the tools you choose. In a photograph of a person's face, for example, the colours change gradually because of light, shadow and complexion. You need a paint function to create this kind of effect; there's no object that you can select or move the way you can with the drawn square i.e., a painting program allows the user to paint arbitrary swaths using brushes of various sizes, shapes, colour and pattern. More painting program allows placement of such predefined shapes as rectangles, polygon and canvas. Any part of the canvas can be edited at pixel level.

The reason why the differences are important is that, as noted earlier, many different kinds of programs offer different kinds of graphics features at different levels of sophistication, but they tend to specialise in one or the other. For example:

1) Many word processors, like Word, offer a handful of simple drawing functions. They aren't that powerful, but if all you need is a basic illustration made up of simple shapes to clarify a point, they're fine.

2) Some programs specialise in graphics creation. Of these, some are all-purpose programs, like KidPix, which offers both drawing and painting functions. KidPix is targeted specifically at children; it has a simplified interface and lacks the sophisticated functions a professional artist might want.

   Other programs, like Adobe PhotoShop, specialise in painting functions, even though they may include drawing functions as well. Painter is a paint-oriented program that offers highly sophisticated, "natural media" functions that approximate the effects of watercolours or drawing with charcoal on textured paper.

   Other graphics programs, such as Adobe Illustrator, specialise in drawing for professional artists and designers; AutoCAD is used mainly for technical and engineering drawing.

3) Page layout, presentation, multimedia authoring and Web development programs usually contain a variety of graphics functions ranging from the simple to the

complex, but their main purpose is composition, not image creation or editing. That is, they allow you to create or import text and graphics and, perhaps, sound, animation and video.

Most of the graphics features in these types of programs are limited to drawing functions because they assume that you will do more complex work in a program dedicated to other functions (e.g., writing in a word processor, editing photos in a paint program), then import your work to arrange the different pieces in the composition program. (Some multimedia authoring systems, however, also offer painting and drawing functions.)

By the way, the differences in composition programs are mainly in the form of their output: Page layout programs, such as PageMaker and QuarkXPress, are for composing printed pages; presentation and multimedia authoring programs, such as PowerPoint and HyperStudio, are for slide shows and computer displays; and Web development applications, like Netscape Composer, are for, well, Web pages.

4) What if you are going to make a magazine, newspaper, book or maybe a multipage menu for a restaurant. In that case, we need a page layout program. The well known softwares in page layout are:

   a) Quark Express
   b) Page Maker (Adobe)
   c) Indesign (Adobe)
   d) Publisher (Microsoft)

   The Queen of Page Layout is Quark Express, owned by Quark Express and Indesign is the King owned by Adobe and finally there is Microsoft Publisher, which is very easy to use.

5) To Create posters, brochures, business cards, stationary, coffee mug design, cereal boxes, candy wrappers, half gallon jugs of orange juice, cups, or anything else you see in print, most designers are going to use vectorised programs to make these things come to life. Vectors are wonderful because they print extremely well, and you can scale them up to make them large, or scale them down to make them small, and there is no distortion. Adobe Illustrator is the King of Vector Programs, hands down. In Adobe Illustrator, you can create a 12 foot, by 12 foot document. If we are going to make anything that is going to be printed, we are doing it in Illustrator. Anything that you create in Illustrator, and the text you use, will come out great. The thing is, Illustrator is hard to learn. It is not an intuitive program at all. This is because vectors use control points called paths and anchor points. To someone new, they are hard to understand, find, and control. That's another story. If you are making a poster, you would make your logo, artwork and text in Illustrator. You would still manipulate your images in Photoshop, and then, "place" them to the Illustrator.

☞ Check Your Progress 1

1) What are the application areas of Computer Graphics. Write short notes on each.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ………

2) What are the file formats available for Presentation Graphics?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

3) Write the full form of (1) TIFF (2) PNG.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

4) Differentiate between Drawing and Painting.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

5) What is Adobe Illustrator used for?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

6) Give some softwares that are suitable for Page Lay out generators like multipage menu for a restaurant?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

7) Is Powerpoint the only presentation graphics software available? If no, then, name some softwares and their developers otherwise provide features of the Powerpoint softwares.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………

1.3.3 Photo Editing

Photo-editing programs are paint programs—it's just that they include many sophisticated functions for altering images and for controlling aspects of the image, like light and colour balance.

For the most part, any paint program can open and display a digital photo image, but it will probably not offer the range and depth of features that a true photo-editing program like PhotoShop does.

Note: KidPix is a general graphics program and PhotoShop is an image-editing program. PhotoShop is the standard used by almost all professional artists

and image editors. Key graphic application involves image editing or manipulation, No matter what type of design you are creating, you are going to manipulate some images. You might change the content of the images, crop, resize, touchup, falsify, fade in, fade out, and/or whatever. Anything that you are going to do to change an image will get done in an image editing or image manipulation application.

There are three big players in image manipulation:

1) PhotoShop (Adobe)
2) FireWorks (Macro Media)
3) Corel (owned by Corel)

Almost everything you see in print or on the web has gone through PhotoShop. It is the king of image manipulation. With PhotoShop you can make anything look real. Photoshop comes bundled with a program called, “ImageReady”.

ImageReady helps your created animated gif, web site rollover effects, image maps and more. Most people that own PhotoShop use less than 10 per cent of its powerful tools.

Fireworks is a super image manipulation application. The thing is, if you open the program, many of the icons, the tool bar, the panels and many of the options in the drop down menus look just like options in PhotoShop. It kind of looks like somebody copied the other persons application.

Note:

- Video editing is in a new and revolutionary stage. Computers really weren’t ready to edit video affordably until right now. Right now, if you have a fast computer and a lot of storage memory, you can create video segments just like anything you see on TV. And, it works well. I would say that the most popular video editing programs are:

  IMovie (apple. Not the best, but easy to use.)
  Adobe Premiere (Adobe)
  Final Cut Pro (Apple)
  Studio Version 9 (Pinnacle Systems)

- **Web Design and Editing**

  To make and edit a website, the big three softwares are:

  1) DreamWeaver (MacroMedia)
  2) Frontpage (MicroSoft)
  3) Go Live (Adobe)
  4) Netscape Composer (Netscape).

Most web developers use DreamWeaver. It is a super tool. It will write your html, css, javascript and create your forms. It is the best.

Frontpage is known for writing lots of code that you don’t need. Go Live is great, but I have never met a person that uses it.

We listed Netscape Composer basically because it is free. It’s not a bad product for free. We teach a lot of people, “Intro do web design,” and if they don’t have any software to make web pages, and if they don’t want to learn html, we show them Composer.

1.3.4 Scientific Visualisation

It is difficult for the human brain to make sense out of the large volume of numbers produced by a scientific computation. Numerical and statistical methods are useful

for solving this problem. Visualisation techniques are another approach for interpreting large data sets, providing insights that might be missed by statistical methods. The pictures they provide are a *vehicle for thinking* about the data.

As the volume of data accumulated from computations or from recorded measurements increases, it becomes more important that we be able to make sense out of such data quickly. Scientific visualisation, using computer graphics, is one way to do this.

Scientific visualisation involve interdisciplinary research into robust and effective computer science and visualisation tools for solving problems in biology, aeronautics, medical imaging, and other disciplines. The profound impact of scientific computing upon virtually every area of science and engineering has been well established. The increasing complexity of the underlying mathematical models has also highlighted the critical role to be played by Scientific visualisation. It, therefore, comes as no surprise that Scientific visualisation is one of the most active and exciting areas of Mathematics and Computing Science, and indeed one which is only beginning to mature. Scientific visualisation is a technology which helps to explore and understand scientific phenomena visually, objectively, quantitatively. Scientific visualisation allow scientists to think about the unthinkable and visualise the unviable. Through this we are seeking to understand data. We can generate beautiful pictures and graphs; we can add scientific information (temperature, exhaust emission or velocity) to an existing object thus becoming a scientific visualisation product.

Thus, we can say scientific visualisation is a scientists tool kit, which helps to simulate insight and understanding of any scientific issue, thus, helping not only in solving or analysing the same but also producing appropriate presentations of the same. This concept of scientific visualisation fits well with modeling and simulation. The *Figure 1* describes steps for visualisation of any scientific problem under consideration, these steps are followed recursively to visualize any complex situation.

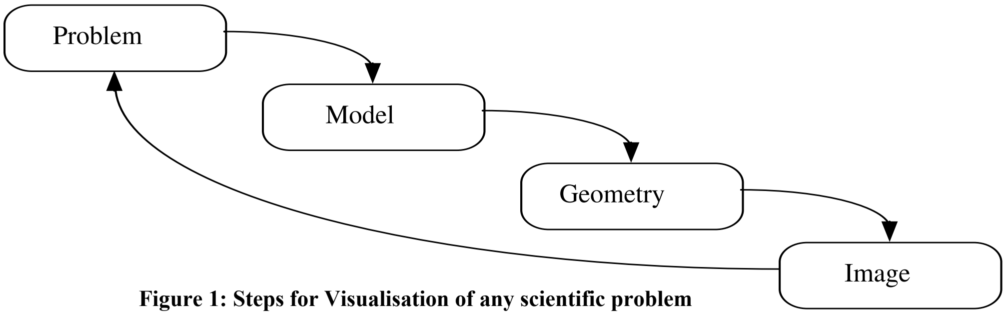


Figure 1: Steps for Visualisation of any scientific problem

Hence, *computer graphics* has become an important part of scientific computing. A large number of software packages now exist to aid the scientist in developing graphical representations of their data. Some of the tools or packages used to express the graphical result for modeling and simulation of any scientific visualisation are:

Matlab (by The Math Works Inc.)

Mathematica or Maple (graphical computer algebra system)

Stella (models dynamic systems)

IDS (Interactive Data Systems) by Research System Inc.

AVS (Application Visualisation System) by Advance visual System Inc.

Excel.

1.3.5 Image Processing

Modern digital technology has made it possible for the manipulation of multi-dimensional signals with systems that range from simple digital circuits to advanced

parallel computers. The goal of this manipulation can be divided into three categories:

1) Image Processing *image in* -> *image out*
2) Image Analysis *image in* -> *measurements out*
3) Image Understanding *image in* -> *high-level description out*

We will focus on the fundamental concepts of *image processing*. We can only make a few introductory remarks about *image analysis* here, as to go into details would be beyond the scope of this unit. *Image understanding* requires an approach that differs fundamentally from the theme of this section. Further, we will restrict ourselves to two-dimensional (2D) image processing although, most of the concepts and techniques that are to be described can be extended easily to three or more dimensions.

We begin with certain basic definitions. An image defined in the "real world" is considered to be a function of two real variables, for example, $a(x,y)$ with a as the amplitude (e.g., brightness) of the image at the *real* coordinate position (x,y). An image may be considered to contain sub-images sometimes referred to as *regions-of-interest*, *ROIs*, or simply *regions*. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions. Thus, one part of an image (region) might be processed to suppress motion blur while another part might be processed to improve colour rendition.

The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantisation process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantised. In other image forming procedures, such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real phase.

A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitisation.

Let us discuss details of digitization. The 2D continuous image $a(x,y)$ is divided into *N rows* and *M columns*. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates $[m,n]$ with $\{m=0,1,2,...,M-1\}$ and $\{n=0,1,2,...,N-1\}$ is $a[m,n]$. In fact, in most cases $a(x,y)$ – which we might consider to be the physical signal that impinges on the face of a 2D sensor – is actually a function of many variables including depth (z), colour (λ), and time (t). The effect of digitisation is shown in *Figure 2*.

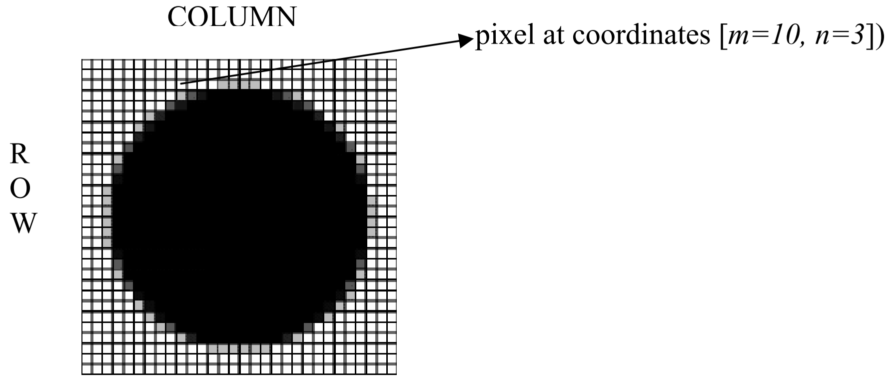


Figure 2: The effect of digitization

The image shown in *Figure 1* has been divided into $N = 30$ rows and $M = 30$ columns for digitisation of a continuous image. The value assigned to every pixel (pixel at coordinates [m=10, n=3]) is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantisation or simply quantisation.

Example Tools and Software for Image Processing

Certain tools are central to the processing of digital images. These include mathematical tools such as *convolution*, *Fourier analysis*, and *statistical* descriptions, and manipulative tools such as *chain codes* and *run codes*. But these tools are worked with at very core levels, in general we use some software to process the image with the help of computers. Some of the categories of image processing software with their respective examples and features are listed below:

1) ***Graphics Image Processing:*** The most commonly used software is: Photoshop.

Features:
- Most common image processing software.
- Focuses on creating a pretty picture.
- Usually limited to popular graphics formats such as: TIFF, JPEG, GIF
- Best suited for working with RGB (3-band) images.
- Does not treat an image as a "map".

2) ***Geographic Information Systems (GIS):*** The most commonly used software is: ArcMap.

Features:
- Works within a geographic context.
- Great for overlaying multiple vector and raster layers.
- It has a somewhat limited analysis although capability, these limitations are being reduced.
- More common than remote sensing software.

3) ***Remote Sensing Packages:*** Commonly used software example is: ERDAS

Features:
- Best suited for satellite imagery.
- Uses geo-spatial information.
- Easily works with multi-spectral data.
- Provides analysis functions commonly used for remote sensing applications.
- Often easy to use but it helps to be familiar with remote sensing.

4) ***Numerical Analysis Packages:*** Commonly used software is: MatLab.

Features:
- Focus usually on numeric processing.
- Programming or mathematical skills usually helpful.
- Used to build more user-friendly applications.

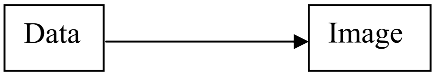
5) ***Web-based Services:*** Commonly used software is: Protected Area Archive.

Features:
- Image display, roam, zoom.
- Image enhancement.
- Simple image processing.

- Distance and area measurement.
- Comparison of old and new images.
- Image annotation (adding text, lines, etc).
- Overlaying vector layers.

Note:

1) Images are the final product of most processes in computer graphics. The ISO (International Standards Organization) defines computer graphics as the sum total of methods and techniques for concerning data for a graphics device by computer, it summarise computer graphics as converting data into images, also called visualisation.



2) Computer graphics concerns the pictorial synthesis of real or imaginary objects from their computer-based models, whereas the related field of image progressing treats the converse process, analysing and reconstruction of sciences. Images processing has sub-areas image enhancement, pattern detection and recognition. This one is used to improve the quality of images by using pattern detection and recognition. OCR is one of example.

1.3.6 Education, Training, Entertainment and Computer Aided Design (CAD)

CAD (or CADD) is an acronym that, depending on who you ask, can stand for:

- Computer Aided Design.
- Computer Aided Drafting.
- Computer Assisted Design.
- Computer Assisted Drafting.
- Computer Assisted Design and Drafting.
- Computer Aided Design and Drafting.

In general acronym for CAD is C*omputer-Aided Design*. In CAD interactive graphics is used to design components and systems of mechanical, electrical, and electronic devices. Actually CAD system is a combination of hardware and software that enables engineers and architects to design everything from furniture to airplanes. In addition to the software, CAD systems require a high-quality graphics monitor; a mouse, light pen or digitised tablets for drawing; and a special printer or plotter for printing design specifications.

CAD systems allow an engineer to view a design from any angle with the push of a button and to zoom in or out for close-ups and long-distance views. In addition, the computer keeps track of design dependencies so that when the engineer changes one value, all other values that depend on it are automatically changed accordingly.

Generally we use CAD as a tool for imparting education and training to the engineers, so that, they can produce beautifully carved and engineered pieces in bulk with the same amount of finishing and perfection. Generally a few terms are used repeatedly with CAD and they are CAM and CNC. Let us discuss **"What are CAD/CAM and CAD/CNC(or NC)"?**

The term CAD/CAM is a shortening of Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM). The term CAD/NC (Numerical Control) is equivalent in some industries.

CAD/CAM software uses CAD drawing tools to describe geometries used by the CAM portion of the program to define a tool path that will direct the motion of a

machine tool to machine the exact shape that is to be drawn on the computer. Let us discuss the terms in brief.

Note:

- **Numerically-Controlled Machines:** Before the development of Computer-aided design, the manufacturing world adopted tools controlled by numbers and letters to fill the need for manufacturing complex shapes in an accurate and repeatable manner. During the 1950s these Numerically-Controlled machines used the existing technology of paper tapes with regularly spaced holes punched in them (think of the paper roll that makes an old-fashioned player piano work, but only one inch wide) to feed numbers into controller machines that were wired to the motors positioning the work on machine tools. The electro-mechanical nature of the controllers allowed digital technologies to be easily incorporated as they were developed. NC tools immediately raised automation of manufacturing to a new level once feedback loops were incorporated (the tool tells the computer where it is, while the computer tells it where it should be).

  What finally made NC technology enormously successful was the development of the universal NC programming language called APT (Automatically Programmed Tools). Announced at MIT in 1962, APT allowed programmers to develop postprocessors specific to each type of NC tool so that, the output from the APT program could be shared among different parties with different manufacturing capabilities.

  Now-a-days many new machine tools incorporate CNC technologies. These tools are used in every conceivable manufacturing sector, like CNC technology is related to Computer Integrated Manufacturing (CIM), Computer Aided Process Planning (CAPP) and other technologies such as Group Technology (GT) and Cellular Manufacturing. Flexible Manufacturing Systems (FMS) and Just-In-Time Production (JIT) are made possible by Numerically-Controlled Machines.

CAD and CAM

The development of Computer-aided design had little effect on CNC initially due to the different capabilities and file formats used by drawing and machining programs. However, as CAD applications such as SolidWorks and AutoCad incorporate CAM intelligence, and as CAM applications such as MasterCam adopt sophisticated CAD tools, both designers and manufacturers are now enjoying an increasing variety of capable CAD/CAM software. Most CAD/CAM software was developed for product development and the design and manufacturing of components and moulds, but they are being used by architects with greater frequency. Thus, a CAD program introduces the concept of real-world measurement. For example, a car or building can be drawn as if it were life-size, and later arranged into sheets and printed on paper at any desired scale.

Note:

1) CAD (or CADD) stands for Computer-Aided Design and Drafting. It differs from both "paint" and "draw" programs in that (i.e., CAD) measurement is central to its abilities. Whereas a "paint" program lets you manipulate each pixel in an array of pixels that make up an image, and a "draw" program goes a step further – it is composed of separate entities or objects, such as circles, lines, etc. It may provide facilities to group these into any object.

2) Is CAD only useful for design drawings?

   No. While true-scale, structurally valid drawings are the reason for CAD's existence, its use is as diverse as our customer's imaginations. For instance, it may be used for:

(a) page layout, web graphics (when scaling and relationships are important to an image, making the image in CAD and exporting it as a bitmap for touchup and conversion can be very productive),

(b) visually accessed databases (imagine a map with detail where you can zoom into an area and edit textual information "in place" and you can then see what other items of interest are "in the neighborhood" - our program's ability to work very rapidly with large drawings is a real plus here),

(c) sign layout, laser-cutting patterns for garment factories, schematic design (where CAD's symbol library capabilities come in handy), and printed-circuit board layout (This was the application that our first CAD program, created in 1977).

Software packages for CAD applications typically provide designer with a multi-window environment. Animations are often used in CAD application, Real-time animations using wire frame displays on a video monitor are useful for testing the performances of a vehicle or a system. The inter frame system allows the user to study the interior of the vehicle and its behaviour. When the study of behaviour is completed, realistic visualising models, surface rendering are used for background scenes and realistic display.

There are many CAD software applications. Some of them with their respective vendors are listed below:

CAD Applications	**Scanner Vendor: Desktop**
AlphaCAM	Canon
Ashlar Vellum	Epson
AutoCAD	Hewlett Packard
CATIA/CADCAM	UMAX

CAD Applications	**Scanner Vendor: Large Format**
Eagle point	Action Imaging
FastCAD	Contex
Pro/E	Xerox
FelixCAD	Kip
IntelliCAD	Ricoh
MasterCAM	Vidar
MasterStation	Widecom

There are many more applications not listed in the list given above.

These applications replicate the old drafting board as a means to draw and create designs. As CAD applications run on computers they provide a great deal more functionality than a drafting board, and a great deal more complexity. The lines and text created in CAD are **vectors**. This means that their shapes and positions are described in mathematical terms. **These vectors are stored on computer systems in *CAD files.***

There are a great **many different file formats for CAD**. Most CAD applications produce their own proprietary file format. The CAD applications from AutoDesk Inc. are used widely. As a result their DWG format is very common. Many other CAD applications from other vendors can produce and open DWG files, as well as their own proprietary formats. CAD data is often exchanged using DXF format.

Note: The DWG file format is a CAD vector format developed by the Autodesk and created by their AutoCAD application. DXF is also a CAD vector format. It is designed to allow the exchange of vector information between different CAD applications. Most CAD applications can save to and read from DXF format.

When CAD drawings are sent to printers the format commonly used is HPGL. HPGL files typically have the extension .plt.

Note: The HPGL file format is a vector format developed by Hewlett Packard for driving plotters. The file extensions used include .plt, .hpg, .hp2, .pl2 and sometimes .prn. However, the use of the .prn extension is not an absolute indicator that the file contains HPGL code. They are often referred to as 'plot files'. Trix Systems offers several options for handling HPGL and the later HPGL2 file formats.

☞ Check Your Progress 2

1) What is Photo Editing? What are the softwares used for image editing?

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………

2) What do you understand by term scientific visualisation, name some software used in this area?

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
…………

3) What is image processing? Give some areas of importance is which the concept of image processing is of use also mention the fruitful softwares in the respective fields.

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
…………

4) Is CAD useful only for designing drawings?

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
…………

5) Briefly discuss DWG, DXF, formats and HPGL files.

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
…………

1.3.7 Simulations

Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analysing the execution output. Simulation embodies the principle of "learning by doing" – to learn about the system we must first build a model of some sort and then operate the model. The use of simulation is an activity that is as natural as a child who *role plays*. Children understand the world around them by simulating (with toys and figures) most of their interactions with other people, animals and objects. As adults, we lose some of this childlike behaviour but recapture it later on through computer simulation. To understand reality and all of its complexity, we must build artificial objects and dynamically act our roles with them. Computer simulation is the electronic equivalent of this type of role playing and it serves to drive synthetic environments and virtual world. Within the overall task of simulation, there are three primary sub-fields: model design, model execution and model analysis (*Figure 3*).

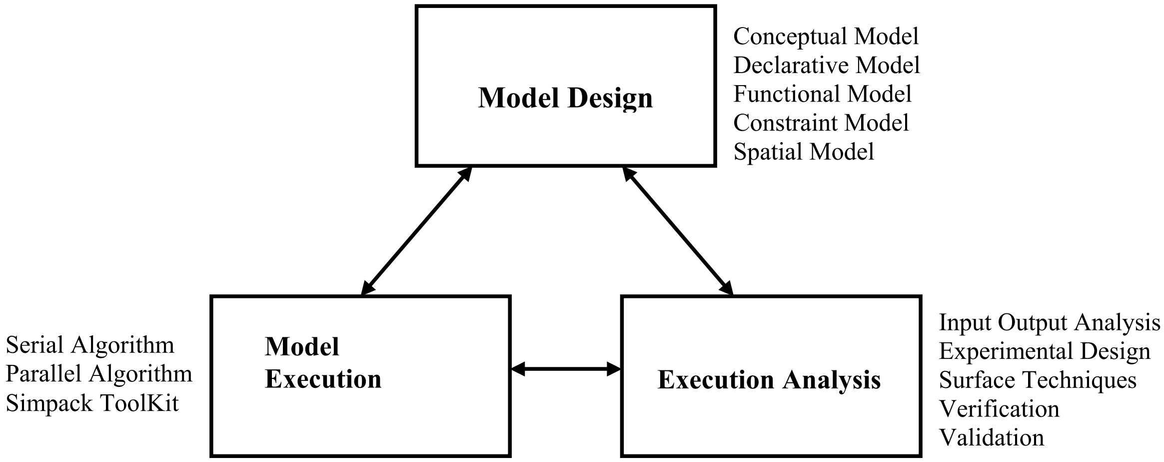


Figure 3: Three Sub-Fields of Computer Simulation

To simulate something physical, you will first need to create a *mathematical model,* which represents that physical object. Models can take many forms including declarative, functional, constraint, spatial or multimodel. A multimodel is a model containing multiple integrated models each of which represents a level of granularity for the physical system. The next task, once a model has been developed, is to execute the model on a computer – that is, you need to create a computer program which steps through time while updating the state and event variables in your mathematical model. There are many ways to "step through time". You can, for instance, *leap* through time using *event scheduling* or you can employ small time increments using *time slicing*. You can also execute (i.e., simulate) the program on a massively parallel computer. This is called *parallel and distributed simulation*. For many large-scale models, this is the only feasible way of getting answers back in a reasonable amount of time.

You may want to know why to do simulation? Is there any other way to do the tasks? To discuss these issues lets briefly discuss the cases in which simulation is essential. There are many methods of modeling systems which do not involve simulation but which involve the solution of a closed-form system (such as a system of linear equations). Let us not go into these issues, as they are not part of our current discussion.

Simulation is often essential in the following cases:

1) The model is very complex with many variables and interacting components;
2) The underlying variables relationships are nonlinear;
3) The model contains random variates;
4) The model output is to be visual as in a 3D computer animation.

The Advantage of Simulation is that – even for easily solvable linear systems – a uniform model execution technique can be used to solve a large variety of systems without resorting to a "bag of tricks" where one must choose special-purpose and sometimes arcane solution methods to avoid simulation. Another important aspect of the simulation technique is that one builds a simulation model to replicate the actual system. When one uses the closed-form approach, the model is sometimes twisted to suit the closed-form nature of the solution method rather than to accurately represent the physical system. A harmonious compromise is to tackle system modeling with a hybrid approach using both closed-form methods and simulation. For example, we might begin to model a system with closed-form analysis and then proceed later with a simulation. This evolutionary procedure is often very effective.

Pitfalls in computer simulation

Although generally ignored in computer simulations, in strict logic the rules governing floating point arithmetic still apply. For example, the probabilistic risk analysis of factors determining the success of an oilfield exploration program involves combining samples from a variety of statistical distributions using the MonteCarlo methods. These include normal, lognormal, uniform and the triangular distributions. However, a sample from a distribution cannot sustain more significant figures than were present in the data or estimates that established those distributions. Thus, abiding by the rules of significant arithmatic, no result of a simulation can sustain more significant figures than were present in the input parameter with the least number of significant figures. If, for instance the net/gross ratio of oil-bearing strata is known to only one significant figure, then the result of the simulation cannot be more precise than one significant figure, although it may be presented as having three or four significant figures.

Note: *Monte Carlo methods* are a widely used class of computational algorithm for simulating the behaviour of various physical and mathematical systems. They are distinguished from other simulation methods (such as molecular dynamics) by being stochastic, that is non-deterministic in some manner – usually by using random number – as opposed to deterministic algorithms. Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to calculation using a computer, utilising many techniques of computer simulation. Further, *Monte Carlo algorithm* is a numerical Monte Carlo method used to find solutions to mathematical problems (which may have many variables) that cannot easily be solved, for example, by integral calculus, or other numerical methods. For many types of problems, its efficiency relative to other numerical methods increases as the dimensions of the problem increases.

1.3.8 Animation and Games

In our childhood, we have all seen the flip books of cricketers which came free along with some soft drink, where several pictures of the same person in different batting or bowling actions are sequentially arranged on separate pages, such that when we flip the pages of the book the picture appears to be in motion. This was a flipbook (several papers of the same size with an individual drawing on each paper so the viewer could flip through them). It is a simple application of the basic principle of physics called persistence of vision. This low tech animation was quite popular in the 1800s when the persistence of vision (which is $1/16^{th}$ of a second) was discovered. This discovery led to some more interesting low tech animation devices like the

zoetrope, wheel of life, etc. Later, depending on many basic mathematics and physics principles, several researches were conducted which allowed us to generate 2d/3d animations. In units 1 and 2 of block 2 we will study the transformations involved in computer graphics but you will notice that all transformations are related to space and not to time. Here lies the basic difference between animation and graphics. The difference is that animation adds to graphics the dimension of time which vastly increases the amount of information to be transmitted, so some methods are used to handle this vast information and these methods are known as animation methods which are classified as:

First Method: In this method, the artist creates a succession of cartoon frames, which are then combined into a film.

Second Method: Here, the physical models are positioned to the image to be recorded. On completion, the model moves to the next image for recording and this process is continued. Thus the historical approach of animation has classified computer animation into two main categories:

(a) *Computer-Assisted Animation* usually refers to 2d systems that computerise the traditional animation process. Here, the technique used is interpolation between key shapes which is the only algorithmic use of the computer in the production of this type of animation equation, curve morphing (key frames, interpolation, velocity control), image morphing.

(b) *Computer Generated Animation* is the animation presented via film or video, which is again based on the concept of persistence of vision because the eye-brain assembles a sequence of images and interprets them as a continuous movement and if the rate of change of pictures is quite fast then it induces the sensation of continuous motion.

This motion specification for computer-generated animation is further divided into 2 categories:

Low Level Techniques (Motion Specific) techniques are used to control the motion of any graphic object in any animation scene fully. Such techniques are also referred as motion specific techniques because we can specify the motion of any graphic object in the scene. Techniques such as interpolation, approximation etc., are used in motion specification of any graphic object. *Low level techniques* are used when animator usually has a fairly specific idea of the exact motion that s/he wants.

High Level Techniques (Motion Generalised) *are techniques* used to describe the general motion behaviour of any graphic object. *These techniques are* algorithms or models used to generate motion using a set of rules or constraints. The animator sets up the rules of the model, or chooses an appropriate algorithm, and selects initial values or boundary values. The system is then set into motion and the motion of the objects is controlled by the algorithm or model. This approach often relies on fairly sophisticated computation such as, vector algebra and numerical techniques among others.

So, the animation concept can be defined as: *A time based phenomenon for imparting visual changes in any scene according to any time sequence. The visual changes could be incorporated through the Translation of the object, scaling of the object, or change in colour, transparency, surface texture etc.*

Note: It is to be noted that computer animation can also be generated by changing camera parameters such as its position, orientation, focal length etc. plus changes in the light effects and other parameters associated with illumination and rendering can produce computer animation too.

Before the advent of computer animation, all animation was done by hand, which involved an enormous amount of work. You may have an idea of the amount of work

by considering that each second of animation film contains 24 frames (film). Then, one can imagine the amount of work in creating even the shortest of animated films. Without going into details of traditional methods let us categorise computer animation technique. Computer animation can be categorised in two ways:

Interactive Computer Animation which is interactively used by users e.g., games. Sprite animation is interactive and used widely in Computer games. In its simplest form it is a 2D graphic object that moves across the display. Sprites often have transparent areas. Sprites are not restricted to rectangular shapes. Sprite animation lends itself well to interactivity. The position of each sprite is controlled by the user or by an application program (or by both). It is called "external" animation.We refer to animated objects (sprites or movies) as "animobs". In games and in many multimedia applications, the animations should adapt themselves to the environment, the program status or the user activity. That is, animation should be *interactive*. To make the animations more event driven, one can embed a script, a small executable program, in every animob. Every time an animob touches another animob or when an animob gets clicked, the script is activated. The script then decides how to react to the event
(if at all). The script file itself is written by the animator or by a programmer. We will discuss about this in Block 4.

Passive Computer Animations: which has no option for users to use computer graphics today is largely interactive e.g., movies. Frame animation is non-interactive animation and is generally used in generating Cartoon movies. This is an "internal" animation method, i.e., it is animation inside a rectangular frame. It is similar to cartoon movies: a sequence of frames that follow each other at a fast rate, fast enough to convey fluent motion. It is typically pre-compiled and non-interactive. The frame is typically rectangular and non-transparent. Frame animation with transparency information is also referred to as "cel" animation. In traditional animation, a cel is a sheet of transparent acetate on which a single object (or character) is drawn.We will discuss this in Block 4.

There are various software which are used to generate computer animations. Some of them are:

- **Flash:** Learning MacroMedia's Flash can be quite complex, but you can do almost anything with it. You can develop presentations, websites, portions of websites, games, or full-length feature, animated cartoons.

  You can import just about anything into Flash. You can drop in images of almost any file format, video clips, sounds and more. It is generally a 2D program.

- **Poser:** Poser by Curious Labs Creates 3D complex models that you can view, from any angle, distance or perspective. You can make the model look like any body you want it to. For instance, if you wanted to make a model that looks just like your Grandmother, you would do it in Poser (the learning curve is vast). Taking that to another level, you could then animate your Grandmother and make her run down a picture of a beach.

There are many more software related to this animation, we will discuss them in the Unit 1 of Block 4.

☞ Check Your Progress 3

1) What do you mean by simulation? What are its uses? Discuss the advantages and pitfalls of simulation.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

……………………………………………………………………………………
……………………………………………………………………………………
…………

2) Differentiate between Graphics and Animation.

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

1.4 GRAPHICS HARDWARE

No matter with which advance graphic software you are working with, if your output device is not good, or hardware handling that software is not good, then ultimate result will be not good, as it could be. We want to say, hardwares also dominate the world of graphics. So, let us discuss some hardware devices which helps us to work with graphic packages.

1.4.1 Input and Output Devices

Input and Output devices are quite important for any software because an inappropriate selection of the concerned hardware may produce some erroneous results or may process data of some other format. So, in the following sections we have planned to discuss some of the input and output devices such as:

- Touch Panel
- Light Pens
- Graphics Tablet
- Plotters
- Film Recorders.

Touch Panels

Touch panels allow displayed object or screen positions to be selected with the touch of the finger and is also known as Touch Sensitive Screens (TSS). A typical application of touch panels is for the selection of processing options that are represented with graphical icons. Touch input can be recorded using optical electrical or acoustical methods.

Optical touch panels employ a line of intra red LEDS (light emitting diodes) along one vertical edge and along one horizontal edge of frame. The opposite vertical and horizontal edges contain light detection. These detections are used to record the beams that may have been interrupted when the panel was touched. Two crossing beams that are interrupted identify the horizontal and vertical coordinates of screen position selected.

An electrical touch panel is constructed with two transparent plates separated by a short distance. One of the plates is coated with a conducting material and the other is resistive material. When the outer plate is touched, it is forced into contact with the inner plate. The contact creates a voltage drop that is converted to a coordinate value of the selected screen position. They are not too reliable or accurate, but are easy to use. Four types are commonly in use. They are as follows:

1) **Electrical TSS:** Wire grid or other conductive coating is utilised to indicate a voltage drop at the point touched point, from which the position may be determined.

2) **Electro-Mechanical TSS:** A glass or plastic sheet with strain gages placed around the edges records the position by the relative magnitude of the deformation of the slightly bent plate.

3) **Optical TSS:** Infrared light from light-emitting diodes (LED) along two perpendicular edges of the screen and detectors along the opposite edges provide an (invisible) optical grid, with reference to which the finger's position is determined.

4) **Acoustic TSS:** Inaudible high-frequency sound waves are emitted along two perpendicular edges and reflected to the emitters by the finger; the echo interval is used as a measure of the distances from the edges.

Light Pen

Light pen is a pointing device. It has a light sensitive tip which is excited when the light is emitted and an illuminated point on the screen comes in its field of view. Unlike other devices which have associated hardware to track the device and determine x and y values, the light pen needs software support (some kind of tracking program). Pointing operations are easily programmed for light pens.

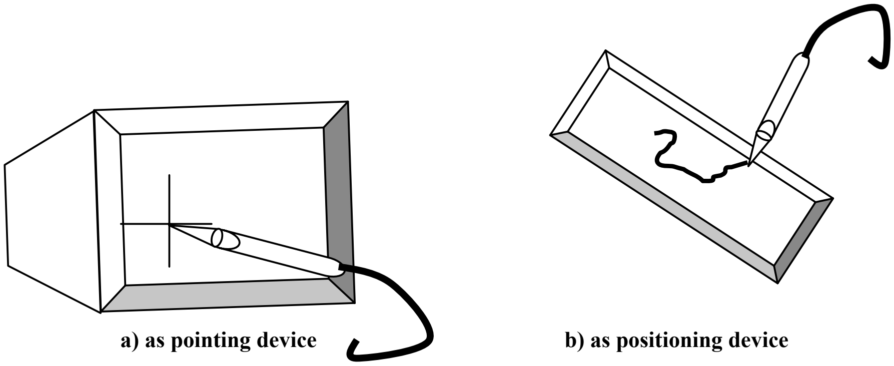


Figure 4: Light Pen Application

Figure 4 shows two typical applications of a light pen. It has a light sensitive tip and a photocell mounted in a pen-like case. If the light pen is pointed at an item on the screen it generates information from which the item can be identified by the program.

When the light pen senses an illuminated phosphor, it interrupts the display processor's interpreting of the file display. The processor's instruction register tells which instruction in the display file was being executed. By identifying the instruction responsible for the illuminated point, the machine can discover which object the pen is pointing to.

A light pen is an **event driven** device. The processor has to wait till it comes across an illuminated point on the screen to obtain any information. The keyboard is another typical example of an event driven device. The processor has to wait for a key to be pressed before it can determine what the user wants to input. Event driven devices can be handled in two ways as follows:

(a) ***Polling:*** The status of each device is periodically checked in a repetitive manner by a polling loop. When an event occurs, the loop is exited and the corresponding event is handled by executing some special event-handling routine or task. Again the polling continues. The disadvantage is that the processor has to wait in an idle state until some event occurs. Data entered can be lost if an event occurs at a time when the main program is not in its polling loop.

(b) ***Interrupts:*** An alternative to polling is the interrupt feature. The device sends an interrupt signal to the processor when an event occurs. The processor breaks from its normal execution and executes some special interrupt-handling routine or task. After the task is complete the control returns to the main program. To handle situations when more than one event occurs, different priorities are assigned to tasks so that higher priority tasks may interrupt tasks of lower priority.

Several events may occur before the program is ready for them. When more than one event occurs, the associate information is entered into the event queue. A polling loop can be employed to check the status of the event queue. The event queue can then pass input data from the polling task to the main program in the correct order. The main program takes events off the head of the queue and invokes the appropriate process. The devices need not be checked repeatedly for occurrence of events. Devices can interrupt even with the processor being unaware of it.

Two kinds of light pen interrupts may occur. If the user points the pen at an item on the screen to select it, as in *Figure 4(a)*, a selection interrupt occurs. If the user is positioning with the pen, as in *Figure 4(b)* a pattern called tracking pattern in displayed along the pen's movement and tracking interrupts occur when the pen sees the tracking pattern.

Modified versions of the light pen may also be used to draw lines, read barcodes, or do transformation operations on objects on the screen (or on a tablet).

Graphics Tablet

Before going into details on the graphic tablet, we need to know what we mean by tablet in computer terminology because, in other disciplines, the word tablet carries different meanings. In terms of computer science "Tablet is a special flat surface with a mechanism for indicating positions on it, normally used as a locator". This small digitiser is used for interactive work on a graphics workstation. Actually this device is essential when someone wants to do free hand drawing or to trace any solid geometrical shape. So a graphic tablet is a drawing tablet used for sketching new images or tracing old ones. Or we may say that a **graphics tablet** is a computer input device that allows one to hand-draw images and graphics, similar to the way one draws images with a pencil on paper. Or a Graphics tablet is a computer peripheral device that allows one to hand images directly to a computer, generally through an imaging program. Graphics tablets consists of a flat surface upon which the user may 'draw' an image using an attached pen-like drawing apparatus using which the user contacts the surface of the tablet, this apparatus is categorised into two types known as pen (or stylus) and puck (a flat block with cross-hairs and some switch keys), which may be wired or wireless. Often mistakenly called a mouse, the puck is officially the "tablet cursor." The image drawn or traced generally does not appear on the tablet itself but rather is displayed on the computer monitor.

The tablet and a hand-held pointer in the form of a stylus (pen) or puck, can serve one or more of these three functions:

(i) For selecting positions (on a drawing or on a menu) on the screen by moving the stylus on the tablet, in a sense using the stylus and tablet as pen on paper.

(ii) For issuing a command or to input a parameter by pressing the stylus at specified pre-programmed locations of a menu on the tablet.

(iii) For digitising the location on a drawing or map placed on the tablet with the stylus or puck.

This device is more accurate and efficient than a light pen. These are two types in use:

(a) **Voltage or Electro-Magnetic Field Tablet and Pointer:** This has a grid of wires, embedded in the tablet surface, with different voltages or magnetic fields corresponding to different coordinates. Intermediate positions within a cell can also be interpolated.

(b) **Acoustic or Sonic (Radio-Wave) Tablet and Pointer:** The sound of a spark at the tip of the stylus is picked up by strip microphones along two edges of the tablet. From the arrival time of the sound pulse at the microphones, the perpendicular distances of the stylus tip from the two axes are known. The acoustic method suffers from its inherent noisiness as well as its susceptibility to interference from other noise.

A combination of electric pulses and time-delay detection by a sensor in the stylus, called **Electro-acoustic Table** is also available.

Tablets typically support two modes of operation:

1) **Digitiser Mode:** creates a one-for-one correspondence between tablet and screen. Wherever you make contact on the tablet, is the exact location on the screen that is affected.

2) **Mouse Mode:** Mouse mode moves the screen pointer relative to any starting position on the tablet surface, just like a mouse.

When drawing or tracing on the tablet, a series of x-y coordinates (vector graphics) are created, either as a continuous stream of coordinates, or as end points. Further the drawings created or traced on tablets are stored as mathematical line segments; and these features of tablets help to produce, tablet computers, tablet PCs and pen tablets.

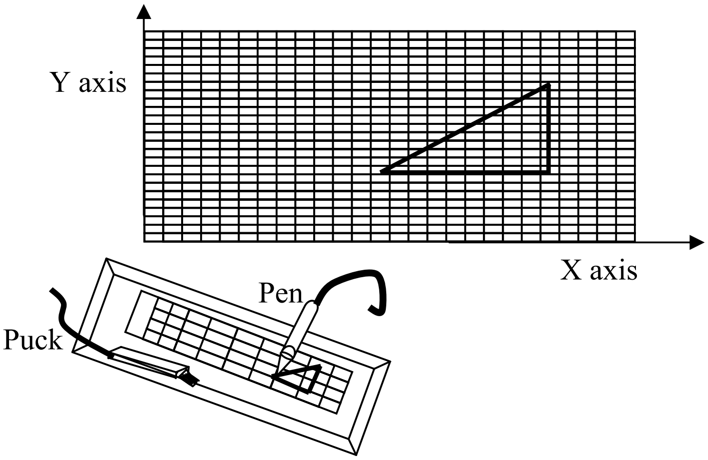


Figure 5: Graphic Tablet

Note: Objects are drawn with a pen (or stylus) or puck, but are traced with the puck only.

Tablet Computer: A complete computer contained in a touch screen. Tablet computers can be specialised for only Internet use or be full-blown, general-purpose PCs with all the bells and whistles of a desktop unit. The distinguishing characteristic is the use of the screen as an input device using a stylus or finger. In 2000, Microsoft began to promote a version of Windows XP for tablet computers, branding them "Tablet PCs".

Pen Tablet: A digitiser tablet that is specialised for handwriting and hand marking. LCD-based tablets emulate the flow of ink as the tip touches the surface and pressure is applied. Non-display tablets display the handwriting on a separate computer screen.

Plotter: A plotter is a vector graphics-printing device that connects to a computer. Now-a-days, we use the plotter right from the field of engineering, to media and advertising. Even in our day-to-day lives we see a large number of computer designed hoardings and kiosks as publicity material. This fine output is achieved by using plotters with computers.

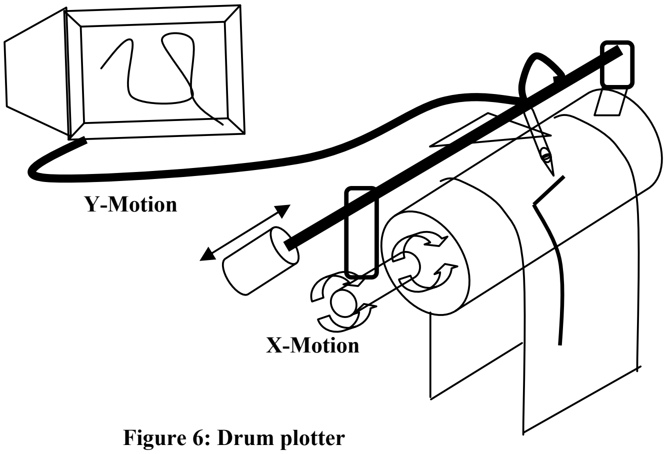


Figure 6: Drum plotter

But the printer may also be connected to the computer. The question then arises, as to how they differ from each other. So let us discuss the differences between them.

1) Plotters print their output by moving a pen across the surface of a piece of paper. This means that plotters are restricted to line art, rather than raster graphics as with other printers. They can draw complex line art, including text, but do so very slowly because of the mechanical movement of the pen.

2) Another difference between the plotter and the printer is that, the printer is aimed primarily at printing text. Thus, the printer is enough to generate a page of output, but this is not the case with the line art of the plotter.

Film Recorders

Film recorder is a graphical output devices for transferring digital images to photographic films.The simplest film recorders typically work by displaying the image on a grayscale Cathode Ray Tube (CRT) placed in front of a photographic camera. For colour images, the red, green, and blue channels are separately displayed on the same grayscale CRT, and exposed to the same piece of film through a filter of the appropriate colour. (This approach yields better resolution and colour quality than one could obtain with a colour CRT). The three filters are usually mounted on a motor-driven wheel. The filter wheel, as well as the camera's shutter, aperture, and film motion mechanism are usually controlled by the recorder's electronics and/or the driving software.

Higher-quality film recorders called LVT (Light Value Transfer) use laser to write the image directly onto the film, one pixel at a time. This method is better suited to print to large-format media such as poster-size prints. In any case, the exposed film is developed and printed by regular photographic chemical processing. Self-developing (polaroid) film can be used for immediate feedback.

Film recorders are used in digital printing to generate master negatives for offset and other bulk printing processes. They are also used to produce the master copies of movies that use computer animation or other special effects based on digital image processing. For preview, archiving, and small-volume reproduction, film recorders have been rendered obsolete by modern printers that produce photographic-quality hardcopies directly on plain paper.

Film recorders were also commonly used to produce slides for slide projectors; but this need is now largely met by video projectors that project images straight from a computer to a screen.

Film recorders were among the earliest computer graphics output devices. Nowadays, film recorders are primarily used in the motion picture film-out process for the ever

increasing amount of digital intermediate work being done. Although significant advances in large venue video projection alleviates the need to output to film, there remains a deadlock between the motion picture studios and theatre owners over who should pay for the cost of these very costly projection systems. This, combined with the increase in international and independent film production, will keep the demand for film recording steady for at least a decade.

☞ Check Your Progress 4

1) What is a graphics tablet? How do the components of a graphic tablet i.e., Pen and Puck differ?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

2) What are touch panels? Discuss different touch panels that are currently available for use?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

3) How does a printer differ from a plotter?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

4) What are file recorders?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

1.4.2 Display Devices

As the importance of input and output devices has been discussed above, let us now focus our discussion specifically on display devices, which present the output to the end user who may not be a technically sound client. If the output display is appealing then your creation will definitely receive a word of appreciation otherwise you may be at the receiving end. Hence, it is pertinent to discuss some of the display devices next.

Refreshing Display Devices

Cathode Ray Tube: It is a refreshing display device. The concept of a refreshing display is depicted pictorially below:

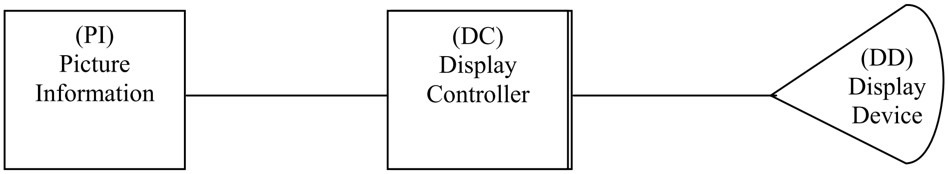


Figure 7: Block Diagram of Display Device

Actually the picture information is given through the stream of electrons (e^-) and its flow is controlled by the display controller (the control is as per the information supplied by the picture) finally the controlled e^- flow produces scintillations on the screen of the display device and the image is formed. The display is known as a refreshing display because display is continuously created by the continuous impugnation/striking of electrons on the screen at the same point. (i.e., same image is continuously made 40-50 times per second i.e., continuous refreshing occurs).

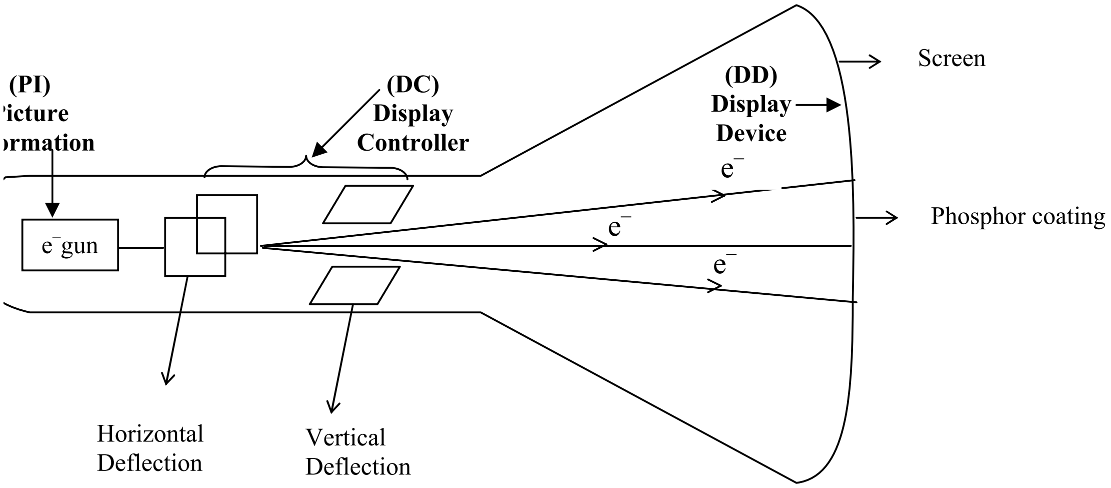


Figure 8: CRT

For proper image we also need to implant a “Digital to Analog converter” (**DAC –** it is an electronic circuit for digital to analog conversion) between the display controller and the display device.

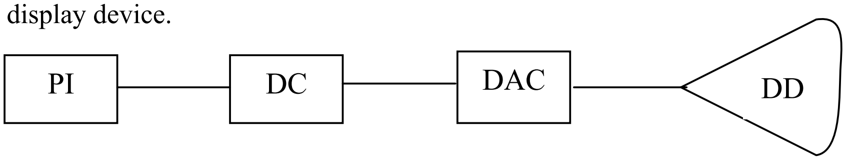


Figure 9: Block Diagram of Display Device with DAC

There are two kinds of refresh monitors, namely, the Random Scan and Raster Scan, which will be described separately.

Note:

1) In a Random Scan System, the Display buffer stores the picture information. Further, the device is capable of producing pictures made up of lines but not of curves. Thus, it is also known as “Vector display device or Line display device or Calligraphic display device”.

2) In a Raster Scan System, the Frame buffer stores the picture information, which is the bit plane (with m rows and n columns).

Because of this type of storage the system is capable of producing realistic images, but the limitation is that, the line segments may not appear to be smooth.

Random Scan Display Device

The original CRT, developed in the late fifties and early sixties, created charts and pictures, line by line on the tube surface in any (*random*) order or direction given, in a vectorial fashion. The electron beam was moved along the particular direction and for the particular length of the line as specified. For this reason, the type of device was known as a **Vector, Calligraphic** or **Stroke** (because it drew lines just as our ancestors drew letters like pictures, stroke by stroke). The process was similar to a hand-sketch or pen-plot.

The graphics commands are transmitted to a display-file program generated by the computer and stored in the **refresh storage area** or buffer memory. The program is executed once in each refresh cycle (of about 1/30 sec.) The electron beam is moved to trace the image line-by-line by a **display processor.** Each line, whether straight or curved, is displayed by the activation of specific points between specified end-point by means of **vector generators** of the analog or digital type, the former being smoother, the latter being faster and cheaper. Curved lines and text characters are displayed as a series of short lines or by a sequence of points.

The display by this system is called **Line Drawing Display.** The sequence operates the following stages, illustrated in *Figure 10*:

1) Graphics Commands
2) Display-File Translator
3) Display-File Program
4) Display (File) Processor
5) VDU

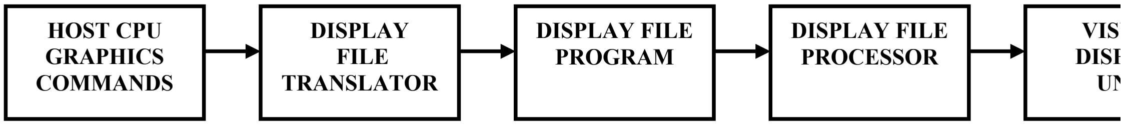
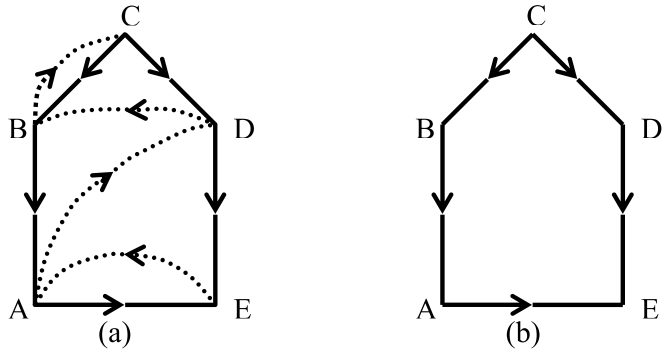


Figure 10: Block Diagram of Line Drawing Display

The process is quite similar to the way a person or a plotter draws a line segment, the pen being moved along a certain direction for a certain length, then changing direction or lifting the pen and moving to a new point, and so on.

For instance, the image of a hut shown in *Figure 11* would be traced as five line segments, the left and right roof slopes, the left and right walls, and the floor line. Efficiency is achieved by minimising wasted motion as would happen if a line segment starts at a point different from the end point of the previous segment – mequivalent to picking up the pen and putting it down at another point to draw the next segment. Thus, it would be better to trace the hut as ABCDE, rather than as CB, CD, BA, DE, and AE, as a draftsman might draw.



a) CB-CD-BA-DE-AE **b) ABCDEA (--- beam off, beam on)**

Figure 11: Vector Scan Display

Raster Scan Display Device

Current day screen display is also based on CRT technology, except that instead of displaying the picture tracing along one vector after another, the image is displayed as a collection of phosphor dots of regular shape arranged in a matrix form. These regular shapes are the pixels (picture elements) and cover the entire screen. The pixels could be (as in earlier times) rectangular, round, or (as is common now) square.
A pixel is the smallest unit addressable by the computer and is made up of a number of smaller dots, comprising the smallest phosphor particles in the CRT coating. However, in this text, we shall use the word "dot" synonymously with "pixel".
The reasons as to why the original CRT did not become popular with the people using computer was because, the refresh procedure required a large memory and high speed, and the equipment to provide these was too expensive. It had to yield to the cheaper storage tube display.

Advances in television technology and chip memories in the mid-seventies overcame both problems with the development of the **raster display**, in which the picture was formed not directly by the lines forming the image but by a choice of appropriate dots from the array of pixels, the entire collection being called the **Raster.** Each row of pixels is called a **Raster Line,** or **Scan Line**.

The electron beam covers the display area horizontally, row by row, from top to bottom in a sweeping motion called **Raster Scan**, each dot lighting up with the intensity and shade of gray or a colour as directed by the Display Controller. Each complete sweep from top left to bottom right of the screen is one complete cycle, called the **Refresh Cycle**.

When viewed from a distance, all the dots together make up the effect of a picture, whether it is a scene from a serial as in the TV or a drawing in computer graphics. The picture looks smooth or coarse-grained depending on the screen **resolution**, that is the number of dots. Typically, on a computer monitor, there are 640 dots in the horizontal direction and 200 to 480 dots in the vertical direction. (The home TV is much finer grained, about three times or more, the scanning being done with analog signals for an entire line at a time, as against the digital information for each dot in the computer monitor). Today's monitors can have resolutions of 1024 by 768 or even higher.

Even with the advent of raster scan, the concept of vector (random scan) graphics has not been completely eliminated. Certain standard geometric shapes such as straight-line segments, circles and ellipses are built into compilers and packages as equations, and developed into pixel graphics for the particular size and parameters specified. Similarly, and more recently, even many fonts in text typography have been reduced to equations so that the input letters, number etc. are really *drawn* from computed lines, as a form of vector graphics. Before going into more details on raster scan display device, let us discuss what is **Raster Scan**. It is the action, which is very similar to that of a dot matrix printer, or even a typewriter that is used to print a pretty border around a message, one line at a time. The image grows from top to bottom, one line at a time, unit completed only when the last line is printed.

The Raster Scan Proceeds as follows: Starting from the top left corner of the screen, the electron gun scans (sweeps) horizontally from left to right, one scan line, that is, one row at a time, jumping (without tracing the line) to the left end of the next lower row until the bottom right corner is reached. Then it jumps (again without tracing) to the top left corner and starts again, finishing one complete refresh cycle. *Figure 12* shows the track of a raster cycle.

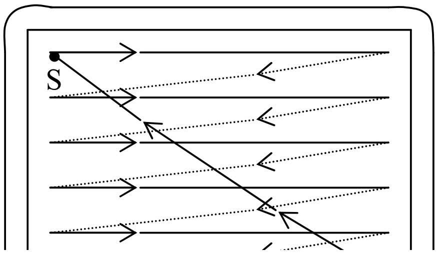


Figure 12: Raster Scan Cycle S-start, E-End

One cycle is completed in about 1/30th of a second, which is faster than the human eye can perceive-thus creating the impression of continuous display and motion.

This technology became very cost-effective, even inexpensive, and because of the availability of large memory and of its high refresh speed, it has become very popular. It takes us back to the refresh-type displays, and especially caters to the interactive and dynamic nature of modern-day computer graphics.

The main disadvantage of the raster scan is the jagged nature of the lines, arising from the fact that the pixels are aligned along regular rows and columns, and points on a line will not, in general, fall on the exact centres of the pixels. But the advantages far outweigh this disadvantage and further developments have diminished this jaggedness problem as well. Hence, almost all the monitors used today have raster display, using pixels, and all subsequent discussions in this text will be concerned with this last type.

All of the preceding characteristics of raster scan will apply to any image on the screen, whether it is graphics proper in the sense of a drawing, or it is a character (letter, number or other symbol).

Three components are necessary for the raster scan display. They are:

1) The **Frame Buffer** which is also called the **Refresh Buffer** or Bitmap. It is the refresh storage area in the digital memory, in which the matrix (array) of intensity values and other parameters (called **attributes**) of all the pixels making up the image are stored in binary form.

2) The display device, which converts the electrical signals into visible images, namely the VDU.

3) The **Display Controller**, the interface that transmits the contents of the frame buffer to the VDU in a form compatible with the display device, a certain number of (30 or more) times a second. The display controller also adjusts and makes allowances for the differences in the operating speeds of the various devices involved, and may also generate line segments and text characters.

Creating points, lines, characters, as well as filling in areas with shades or colours are all accomplished by this scan technique, known as the **Frame Buffer Display.** A common method for storing characters is to store the pixel information for the entire matrix (5*7 to 9*14, horizontal to vertical) assigned to represent a character.

The sequence of operations here is by the following stages, as depicted in *Figure 13.*

1) Graphics Commands
2) Display Processor (Scan Conversion)
3) Frame Buffer
4) Display Controller
5) VDU

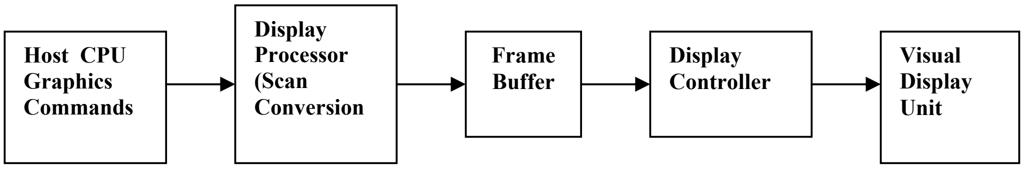


Figure 13: Block diagram of Raster Scan Display Device

Frame Buffers

The storage area in a raster scan display system is arranged as a two-dimensional table. Every row-column entry stores information such as brightness and/or colour value of the corresponding pixel on the screen. In a frame buffer each pixel can be represented by 1 to 24 or more bits depending on the quality (resolution) of the display system and certain attributes of the pixel. Higher the resolution, better the quality of the pictures. Commands to plot a point or line are converted into intensity and colour values of the pixel array or bitmap of an image by a process called **Scan Conversion**.

The display system cycles through the refresh buffer, row-by-row at speeds of 30 or 60 times per second to produce the image on the display. The intensity values picked up from the frame buffer are routed to the Digital/Analog converter which produces the necessary deflection signals to generate the raster scan. A flicker-free image is produced by interlacing all odd-numbered scan lines that are displayed first from, top to bottom and then, all even-numbered scan lines that are displayed. The effective refresh rate to produce the picture becomes much greater than 30 Hz.

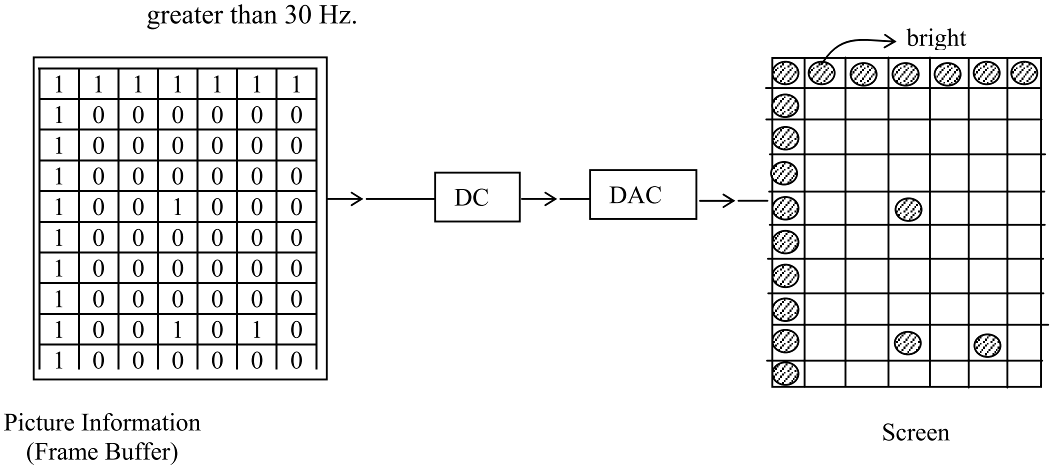


Figure 14: If information stored in frame buffer is 1 then, the corresponding pixel is made bright on the screen and if it is zero then no brightness appears i.e., 0→off; 1 → ON so the image obtained on the screen is discrete.

Different kinds of memory have been used in frame buffers: The earliest type of frame buffers used drums and disks with rotational frequency compatible to the rate of refresh. Such frame buffers were called **rotating-memory frame buffers**. But the relatively lower costs of integrated-circuit shift registers saw the rotating-memory frame buffer being replaced by the **shift-register frame buffers**.

A frame buffer can be constructed with a number of shift registers, each representing one column of pixels on the screen. Each shift register contributes one bit per horizontal scan line. However, changing a given spot on the screen is not very easy with shift registers. So they are not suitable for interactive applications.

Modern frame buffers use random-scan integrated circuits (discussed earlier) where the pixel intensities are represented by 1,2,4,8,16 or 24 bits. Encoding text and simple images does not require more than say, 8 bit per pixel. But to produce a good quality coloured image more than 8 bits, something like 24 bits, are required.

One of the best methods to encode coloured pictures involves the use of a colour map. The pixel values in the frame buffer are treated as addresses of a look-up-table,

which has entries for every pixel's red, green and blue components. The entry value is used to control the intensity or colour on the CRT; each of the colour components can be defined to high precision providing accurate control over the colours displayed.

Another type of frame buffer is the **multiple-plane frame buffer,** where the frame buffer can be treated as consisting of several frames or planes, each containing information (intensity and/or colour) values of a separate image. An 8-bit per pixel frame buffer can be made to represent a single image with 8-bits of intensity precision or it can represent tow images each of 4-bit intensity precision or eight black and white images with 1-bit intensity precision each. A variety of image mixing can be done. For example, in animation systems, several moving objects can be displayed as separate planes.

Note:

1) In a frame buffer, information storage starts from top left corner and goes till the bottom right corner.
2) Using this type of buffer we can draw curves too.
3) So in order to draw live images of objects of day-to-day life, enormous picture information is required and this is the limitation.
4) How to vary intensity of a point (bright point) in Raster scan display device)

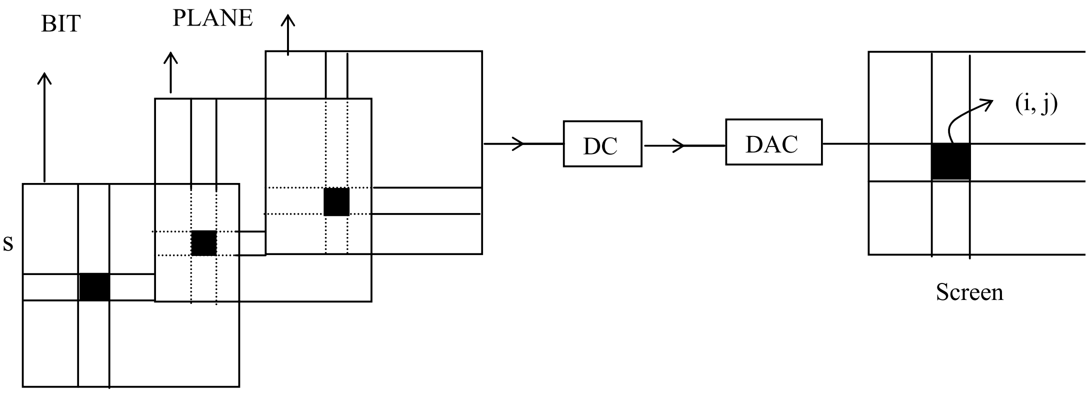


Figure 15: Frame buffer: Intensity Variation of a pixel

Picture Information

Here, the picture information is stored in the form of bit plans (on each bit plane full information of picture is stored) and for brightest intensity 3 bit plane (say for simplicity) should have 1 as respective information in the frame buffer, if it is zero then, the intensity will be the least; some of the intensity combination are discussed below:

bit plane			
A	B	C	
0	0	0	⟶ min. intensity of a point
0	0	1	
0	1	0	⟶ value of a pixel in 2nd bit plane
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	⟶ max intensity of a point

Maximum intensity of a pixel implies that the value for that point on each bit plane is 1 which generates maximum intensity (to have a bit low intensity any one of the bit plane is given information 0 for that point and so on):

If picture information is stored in 2 bit planes then possibilities are:

0 0 ⟶ min. intensity of a point
0 1 ⟶ medium intensity of a point
1 0 ⟶ medium intensity of a point
1 1 ⟶ max intensity of a point

For corresponding pixel on screen:

- each information digit is 0 or 1 for respective bit plane in continuation,
- each digit constitutes the information for a point in the respective bit plane,
- for 2 bit planes we have 4 levels, similarly, 3 bit planes we have 8 levels i.e., for n bit planes we have 2^n levels.
  ⇒ for n bit plane 2^n amount of information is needed.
- By adopting this method of picture information storage, no doubt, we can vary the intensity of adopting but more memory consumption occurs because there exists more bit planes.
- For controlling the intensity of the colours, we need a minimum of 8 bit planes i.e., picture information has to have 8 digits (0, 1) further 2^8 combinations occur.
- For colours case we need in general 3 electron guns of 3 colours (red, green and blue) for picture display with varying colours.

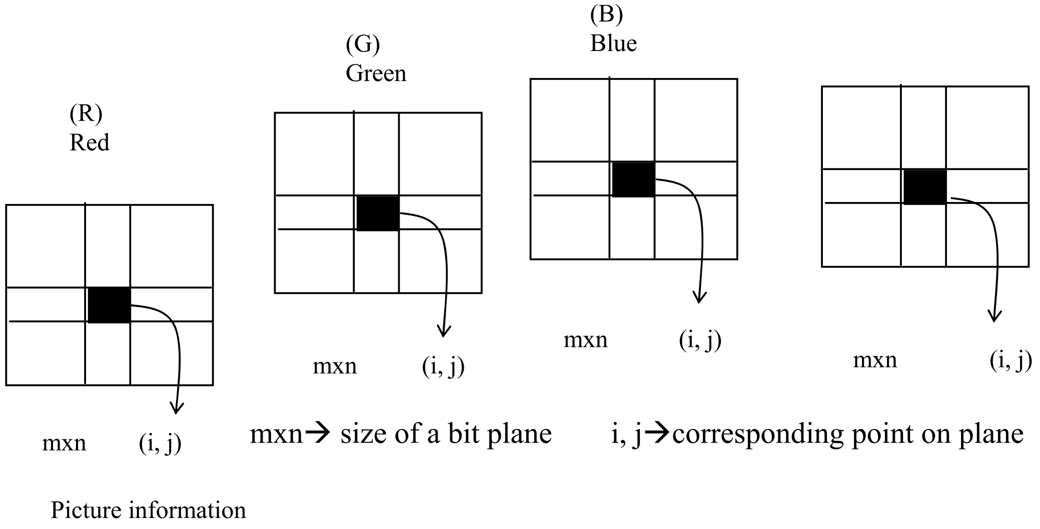


Figure 16: Frame Buffer: Colour Variation of a Pixel

As in the figure above we have 3 bit planes so we have 3 digit numbers. If information in each plane (R, G, B) is zero i.e., (0, 0, 0) then there is no display. Other situations are listed below:

	R	G	B	
So,	0	0	0	no display
	0	0	1	Blue display
	0	1	0	Green display
	0	1	1	Green-blue mix display and so on.

Similarly, for more colours we should have more bit planes and hence more numbers of digits signify more colour combinations.

Similarly we can control the intensity of colours

We have more bit planes for each colour say 3 bit planes for R, 3 for G and 3 for B (each bit plane of size mxn) so there exist a digit number and by respective 1s and 0s we can control colour intensity too. Here, the amount of information needed is 2^9 or amount of information needed is 2^3, 2^3, 2^3 for each R, G, B. Thus we can generate light red, light green or combination of light red, dark green, medium blue and so on.

Example 1: What is the number of memory bits required for 3 bit –plane frame buffer for a 512 x 512 raster.

Solution: Total memory bits required are 3 x 512 x 512 = 786,432

Example 2: What is the refresh rate in a 512 x 512 raster if pixels are accessed at the rate of 200 nanoseconds.

Solution. For individual access of pixels rate is 200×10^{-9} seconds, for 512x512 pixels 0.0524 seconds required.

Refresh rate per second is 1/0.0524 =19 frames/seconds.

Plasma Panel

It is an inherent memory device. Images can be written onto the surface point by point and they remain stable, without flicker, for a long time after being intensified. An inert mixture of noble gases (neon and xenon) gas is filled and sealed between two glass sheets with fine-mesh gold-wire electrodes attached to their inner faces. The particles of the gas behave in a bi-stable manner, that is stable at two levels (on/off). When voltage is applied across the electrodes the gas molecules get ionised and are activated giving rise to a glow.

The fine mesh of electrodes makes up thousands of addressable cells. A definite picture is generated when gas particles in the corresponding cells are excited by the application of the appropriate voltages. The ionised gas molecules in the cells excited by the applied voltages glow, making the picture visible until the current is turned off. This gives a steady image but the resolution is poor (60 dots per inch), and the hardware is much more complex (hence costlier) than raster scan.The plasma display panel is less bulky than the CRT but also vary the cost of construction is very high. Addressing of cells and wiring is complex.

Advantage

- Slim design (Wall mountable)
- Larger than LCD screens

Disadvantage

- Expensive, although cheaper than LCDs in larger sizes.
- Is subject to screen burn-in, but modern panels have a manufacturer rated lifespan of 50,000 or more hours.
- First 2000 hours is its brightest point. Every hour thereafter, the display gradually dims.
- At higher elevations, usually 6000 ft or higher, they exhibit noticeable humming.

LCD

Liquid Crystal Display is a type of display used in digital watches and many portable computers. These work with polarised ambient (outside source) light consisting of liquid crystal (a material which polarises light when a voltage is applied to it), with a

conductive coating for the voltage application, set between two transparent glass or plastic plates and a polarised film on one side which will show the excited portions of the liquid crystal as dark dots or lines. (The seven-segment display of most digital watches is an example of LCD by lines). This technology is now applied to Data Projectors to project computer generated or stored images directly on to big screens in auditoriums.

LCD displays utilise two sheets of polarising material with a liquid crystal solution between them. An electric current passed through the liquid causes the crystals to align so that light cannot pass through them. Each crystal, therefore, is like a shutter, either allowing light to pass through or blocking the light.

Monochrome LCD images usually appear as blue or dark gray images on top of a grayish-white background. Colour LCD displays use two basic techniques for producing colour: *Passive matrix* is the less expensive of the two technologies. The other technology, called *Thin Film Transistor* (TFT) or *active-matrix*, produces colour images that are as sharp as traditional CRT displays, but the technology is expensive. Recent passive-matrix displays using new CSTN and DSTN technologies produce sharp colours rivaling active-matrix displays. Most LCD screens used in notebook computers are backlit, or transmissive, to make them easier to read.

☞ Check Your Progress 5

1) What are Refreshing display devices? What are their limitations?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

………………

2) Differentiate between Random and Raster Scan display devices.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………

1.5 SUMMARY

In this unit, we have discussed the conceptual meaning of computer graphics, with its application in various fields right from presentation graphics to animation and games. We have also discussed the variety of software and their respective file formats used in various applications of computer graphics. In the end, we have discussed the working of various input and output devices. Finally, the discussion on display devices was done with coverage to Refreshing Display Devices, and Plasma Panels, and LCD Display Devices.

1.6 SOLUTIONS/ANSWERS

Check Your Progress 1

1) A few of the various applications areas which are influenced by Computer graphics are:

- Presentation Graphics
- Painting Drawing
- Photo Editing
- Scientific Visualisation
- Image Processing
- Digital Art
- Education, Training, Entertainment and CAD
- Simulation
- Animation and games.

2) GIF, JPG, BMP, PSD, TIFF, PNG etc.

3) TIFF or PNG: TIFF has been around longer than PNG, which was originally designed to replace GIF on the Web. PowerPoint works well with both of these files when creating transparent backgrounds but generally PNG creates smaller file sizes with no loss of quality.

4)

Drawing	***Painting***
Drawing is a software application means using tools that create "objects," such as squares, circles, lines or text, which the program treats as discrete units. If you draw a square in PowerPoint, for example, you can click anywhere on the square and move it around or resize it. It's an object, just like typing the letter "e" in a word processor, i.e., a drawing program allows a user to position standard shape (also called symbols, templates, or objects) which can be edited by translation, rotations and scaling operations on these shapes.	***Painting*** functions, on the other hand, don't create objects. If you look at a computer screen, you'll see that it's made up of millions of tiny dots called pixels. You'll see the same thing in a simpler form if you look at the colour comics in the Sunday newspaper—lots of dots of different colour ink that form a picture. Unlike a drawing function, a paint function changes the colour of individual pixels based on the tools you choose. In a photograph of a person's face, for example, the colours change gradually because of light, shadow and complexion. You need a paint function to create this kind of effect; there's no object that you can select or move the way you can with the drawn square, i.e., a painting program allows the user to paint arbitrary swaths using a brush various size, shape, colour and pattern. More painting programs allows placement of such predefined shapes as rectangles, polygon and canvas. Any part of the canvas can be edited at the pixel level.

The reason why the differences are important is that, as noted earlier, many different kinds of programs offer different kinds of graphics features at different levels of sophistication, but they tend to specialise in one or the other.

5) To Create posters, brochures, business cards, stationary, coffee cup mug design, cereal boxes, candy wrappers, orange juice gallon jugs, cups, or anything else you see in print. Most designers will use vectorised programs to make these things come to life. Vectors are wonderful because they print extremely well, and you can scale them up to make them large, or scale them down to make them small, and there is no distortion. **Adobe Illustrator is the King of Vector Programs**. In Adobe Illustrator, you can create a 12 foot, by 12 foot document.

6) If you are going to make a magazine, newspaper, book or maybe a multipage menu for a restaurant. In that case, we need a page layout program. The well known softwares in page layout are:

   (a) Quark Express
   (b) Page Maker (Adobe)
   (c) Indesign (Adobe)
   (d) Publisher (MicroSoft)

7) No, other example of softwares are Apple's Keynote, Openoffice's (Star Office-by Sun microsystems), Impress, Microsoft Powerpoint and (for multimedia presentations, incorporating moving pictures, and sounds) Macromedia Director. Custom graphics can also be created in other programs such as Adobe Photoshop or Adobe Illustrator.

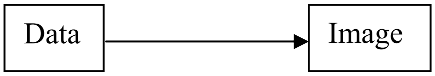
Check Your Progress 2

1) Photo-editing stream involves programs, which are not just paint programs—but they include many sophisticated functions for altering images and for controlling aspects of the image, like light and colour balance. Some of the professionally used software for photo editing are PhotoShop (Adobe), FireWorks (Macro Media), Corel (owned by Corel) etc.

2) Scientific visualisation involves interdisciplinary research into robust and effective computer science and visualisation tools for solving problems in biology, aeronautics, medical imaging, and other disciplines. The profound impact of scientific computing upon virtually every area of science and engineering has been well established. Some examples of the software used in this field are:

   Matlab (by The Math Works Inc.)
   Mathematica or Maple (graphical computer algebra system)
   Stella (models dynamic systems)
   IDL (Interactive Data Systems) by Research System Inc.
   AVS (Application Visualisation System) by Advance visual System Inc.

3) Image Processing means *image in* → processed *image out*
   Images are the final product of most processes in computer graphics. The ISO (International Standards Organisation) defines computer graphics as the sum total of methods and techniques for concerning data for a graphics device by a computer. It summarise and computer graphics as converting data into images, which is known as visualisation.



   Some of the categories of image processing software with their respective examples and features are listed below:

   (a) ***Graphics Image Processing:*** Commonly used software example is: Photoshop.
   (b) ***Geographic Information Systems (GIS):*** Commonly used software example is: ArcMap.
   (c) ***Remote Sensing Packages:*** Commonly used software example is: ERDAS.
   (d) ***Numerical Analysis Packages:*** Commonly used software example is: MatLab.

(e) ***Web-based Services:*** Commonly used software example is: Protected Area Archive.

4) No. While true-scale, structurally valid drawings are the reason for CAD's existence, its use is as diverse as our customer's imaginations.

   (a) Page layout, web graphics (when scaling and relationships are important to an image, making the image in CAD and exporting it as a bitmap for touchup and conversion can be very productive).
   (b) Visually accessed databases (imagine a map with details where you can zoom into an area and edit textual information "in place" and you can then see what other items of interest are "in the neighbourhood" – our program's ability to work very rapidly with large drawings is a real plus here).
   (c) Sign layout, laser-cutting patterns for garment factories, schematic design (where CAD's symbol library capabilities come in handy), and printed-circuit board layout (This was the application that our first CAD program, created in 1977).

5) The DWG file format is a CAD vector format developed by Autodesk and created by their AutoCAD application. DXF is also a CAD vector format. It is designed to allow the exchange of vector information between different CAD applications. Most CAD applications can save to and read from DXF format.

   When CAD drawings are sent to printers the format commonly used is HPGL. HPGL files typically have the extension .plt.

   The HPGL file format is a vector format developed by Hewlett Packard for driving plotters. The file extensions used include .plt, .hpg, .hp2, .pl2 and sometimes .prn. However, the use of the .prn extension is not an absolute indicator that the file contains an HPGL code. They are often referred to as 'plot files'. Trix Systems offers several options for handling HPGL and the later HPGL2 file formats.

Check Your Progress 3

1) Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analysing the execution output. Simulation embodies the principle of "learning by doing" – to learn about the system we must first build a model of some sort and then operate the model. Simulation is often essential in the following cases:

- the model is very complex with many variables and interacting components;
- the underlying variables relationships are nonlinear;
- the model contains random variates;
- the model output is to be visual as in a 3D computer animation.

The Advantage of Simulation is that – even for easily solvable linear systems – a uniform model execution technique can be used to solve a large variety of systems without resorting to a "bag of tricks" where one must choose special-purpose and sometimes arcane solution methods to avoid simulation. Another important aspect of the simulation technique is that one builds a simulation model to replicate the actual system. When one uses the closed-form approach, the model is sometimes twisted to suit the closed-form nature of the solution method rather than to accurately represent the physical system. A harmonious compromise is to tackle system modeling with a hybrid approach using both closed-form methods and simulation. For example, we might begin to model a system with closed-form analysis and then proceed later with a simulation. This evolutionary procedure is often very effective.

Pitfalls in Computer Simulation

Although generally ignored in computer simulations, in strict logic the rules governing floating point arithmetic still apply. For example, the probabilistic risk analysis of factors determining the success of an oilfield exploration program involves combining samples from a variety of statistical distributions using the MonteCarlo methods. These include normal, lognormal, uniform and the triangular distributions. However, a sample from a distribution cannot sustain more significant figures than were present in data or estimates that established those distributions. Thus, abiding by the rules of significant arithmatic, no result of a simulation can sustain more significant figures than were present in the input parameter with the least number of significant figures. If, for instance the net/gross ratio of oil-bearing strata is known to only one significant figure, then the result of the simulation cannot be more precise than one significant figure, although it may be presented as having three or four significant figures.

2) Animation is a time based phenomenon for imparting visual changes in any scene according to any time sequence, the visual changes could be incorporated through translation of object, scaling of object, or change in colour, transparency, surface texture etc., whereas Graphics does not contain the dimension of time.

Graphics + Dimension of Time = Animation

Check Your Progress 4

1) A graphic tablet is a drawing tablet used for sketching new images or tracing old ones. Or we can say that a graphics tablet is a computer input device that allows one to hand-draw images and graphics, similar to the way one draws images with a pencil or paper. Or Graphics tablet is a computer peripheral device that allows one to hand images directly into a computer, generally through an imaging program.

   Difference between pen and puck: Objects are drawn with a pen (or stylus) or puck, but are traced with the puck only.

2) Touch panels allow displayed objects or screen positions to be selected with the touch of the finger, also known as Touch Sensitive Screens (TSS). Four types of touch panels commonly in use, are Electrical, Electro-Mechanical, Optical, Acoustic touch panels.

3) The differences between printers and plotters:

   (a) Plotters print their output by moving a pen across the surface of piece of a paper. This means that plotters are restricted to line art, rather than raster graphics as with other printers. They can draw complex line art, including text, but do so very slowly because of mechanical movement of pen.

   (b) Another difference between plotter and printer is that the printer is aimed primarily at printing text. Thus, the printer is enough to generate a page of output, but this is not the case with the line art on a plotter.

4) Film recorder is a graphical output devices for transferring digital images to photographic films.

Check Your Progress 5

1) Refreshing display devices are those display devices in which the picture continuously refreshes. The general refresh rate is 40 to 60 times per second example CRT. There are two kinds of refreshing display devices, namely the random scan and raster scan, the limitation is that these devices are not competent to provide smooth and good quality images.

2) In Random Scan system the Display buffer stores the picture information, further the device is capable of producing pictures made up of lines but not of curves. Thus, it is also known as “Vector display device or Line display device or Calligraphic display device.

   In Raster Scan system the Frame buffer stores the picture information which is the bit plane (with m rows and n columns) because of this type of storage the system is capable of producing realistic images, but the limitation is that the line segments may not appear to be smooth.

UNIT 2 GRAPHIC PRIMITIVES

Structure **Page Nos.**

2.1 Introduction 46
2.2 Objectives 46
2.3 Points and Lines 46
2.4 Line Generation Algorithms 48
2.4.1 DDA Algorithm 49
2.4.2 Bresenhams Line Generation Algorithm 54
2.5 Circle-Generation Algorithms 59
2.5.1 Properties of Circle 59
2.5.2 Mid Point Circle Generation Algorithm 61
2.6 Polygon Filling Algorithm 63
2.7 Summary 68
2.8 Solutions/Answers 68

2.1 INTRODUCTION

In unit 1 of this block, we have discussed refreshing display devices and its types that are Random Scan display devices and Raster Scan display devices. We have also discussed the limitations of these display devices. In Raster Scan display device, which we have discussed in the previous unit, the picture information is stored in the frame buffer, the concept of frame buffer conveys that the information for the image to be projected on the screen is stored in the form of 0s and 1s, making respective pixels activate and deactivate on the screen, and it is the concept itself which contributes to the discreteness in the picture under display. So, in this unit we are going to extend our discussion to algorithms, which are responsible for actual display of the geometries like line, circle etc., on display devices, such that there is decrease in discreteness and hence, more realistic finished image is the outcome.

2.2 OBJECTIVES

After going through this unit, you should be able to:

- describe the Line Generation Algorithm;
- apply Line Generation Algorithm to practical problems;
- describe the different types of Line Generation Algorithm;
- describe Circle Generation Algorithm;
- apply Circle Generation algorithm to practical problems;
- describe the different types of Circle Generation Algorithms;
- describe Polygon fill algorithm, and
- describe Scan Line Polygon fill algorithm.

2.3 POINTS AND LINES

In the previous unit, we have seen that in order to draw primitive objects, one has to first scan convert the objects. This refers to the operation of finding out the location of pixels to be intensified and then setting the values of corresponding bits, to the desired intensity level. Each pixel on the display surface has a finite size depending on the screen resolution and hence, a pixel cannot represent a single mathematical point. However, we consider each pixel as a unit square area identified by the coordinate of its lower left corner, the origin of the reference coordinate system being located at the

lower left corner of the display surface. Thus, each pixel is accessed by a non-negative integer coordinate pair (*x*, *y*). The x values start at the origin and increase from left to right along a scan line and the y values

(i.e., the scan line numbers) start at bottom and increase upwards.

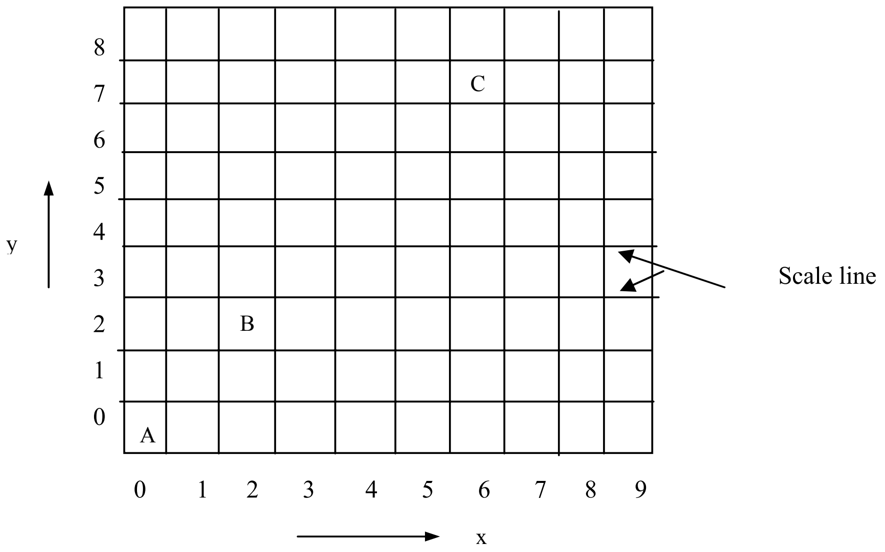


Figure 1: Scan lines

Figure 1, shows the Array of square pixels on the display surface. Coordinate of pixel A: 0, 0; B: 2, 2; C: 6, 7. A coordinate position (6.26, 7.25) is represented by C, whereas (2.3, 2.5) is represented by B. Because, in order to plot a pixel on the screen, we need to round off the coordinates to a nearest integer. Further, we need to say that, it is this rounding off, which leads to distortion of any graphic image.

Line drawing is accomplished by calculating the intermediate point coordinates along the line path between two given end points. Since, screen pixels are referred with integer values, plotted positions may only approximate the calculated coordinates – i.e., pixels which are intensified are those which lie very close to the line path if not exactly on the line path which is in the case of perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line, we will discuss such algorithms in our next section. Screen resolution however, is a big factor towards improving the approximation. In a high resolution system the adjacent pixels are so closely spaced that the approximated line-pixels lie very close to the actual line path and hence, the plotted lines appear to be much smoother — almost like straight lines drawn on paper. In a low resolution system, the same approximation technique causes lines to be displayed with a “stairstep apperance” i.e., not smooth as shown in *Figure 2*, the effect is known as the stair case effect. We will discuss this effect and the reason behind this defect in the next section of this unit.

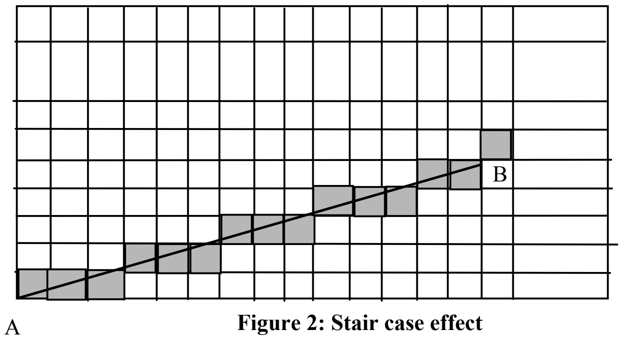


Figure 2: Stair case effect

2.4 LINE GENERATION ALGORITHMS

- In unit 1, we have discussed the case of frame buffer where information about the image to be projected on the screen is stored in an *m*n* matrix, in the form of 0s and 1s; the 1s stored in an *m* n* matrix positions are brightened on the screen and 0's are not brightened on the screen and this section which may or may not be brightened is known as the Pixel (picture element). This information of 0s and 1s gives the required pattern on the output screen i.e., for display of information. In such a buffer, the screen is also in the form of *m* n* matrix , where each section or niche is a pixel (i.e., we have *m* n* pixels to constitute the output).

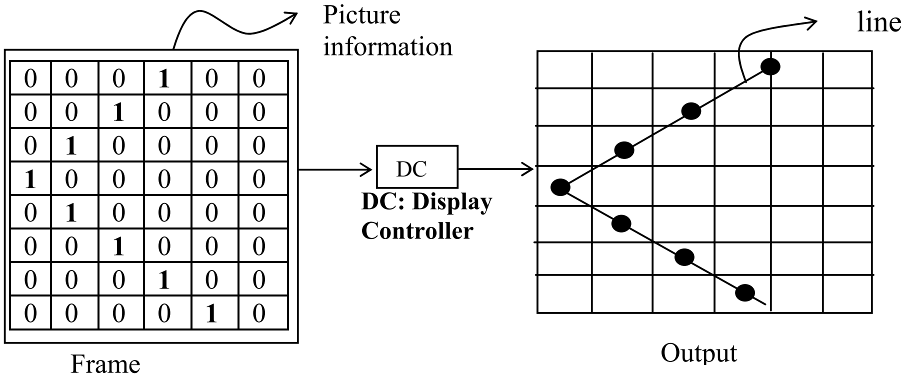


Figure 3: Basic Line Generation

Now, it is to be noted that the creation of a line is merely not restricted to the above pattern, because, sometimes the line may have a slope and intercept that its information is required to be stored in more than one section of the frame buffer, so in order to draw or to approximate such the line, two or more pixels are to be made ON. Thus, the outcome of the line information in the frame buffer is displayed as a stair; this effect of having two or more pixels ON to approximating a line between two points say A and B is known as the Staircase effect. **The concept is shown below in *Figure 4.***

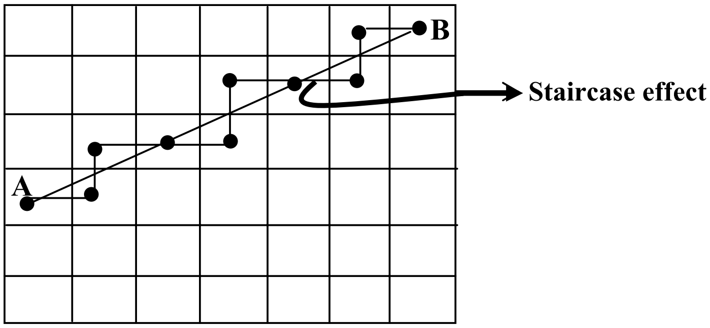


Figure 4: Staircase effect

So, from the *Figure 4*, I think, it is clear to you that when a line to be drawn is simply described by its end points, then it can be plotted by making close approximations of the pixels which best suit the line, and this approximation is responsible for the staircase effect, which miss projects the information of the geometry of line stored in the frame buffer as a stair. This defect known as Staircase effect is prominent in DDA Line generation algorithms, thus, to remove the defect Bresenham line generation Algorithm was introduced. We are going to discuss DDA (Digital Differential Analyser) Algorithm and Bresenham line generation Algorithm next.

2.4.1 DDA (Digital Differential Analyser) Algorithm

From the above discussion we know that a Line drawing is accomplished by calculating intermediate point coordinates along the line path between two given end points. Since screen pixels are referred with integer values, or plotted positions, which may only approximate the calculated coordinates – i.e., pixels which are intensified are those which lie very close to the line path if not exactly on the line path which in this case are perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line, one such algorithm is the DDA (Digital Differential Analyser) algorithm. Before going to the details of the algorithm, let us discuss some general appearances of the line segment, because the respective appearance decides which pixels are to be intensified. It is also obvious that only those pixels that lie very close to the line path are to be intensified because they are the ones which best approximate the line. Apart from the exact situation of the line path, which in this case are perfectly horizontal, vertical or 45° lines (i.e., slope zero, infinite, one) only. We may also face a situation where the slope of the line is > 1 or < 1.Which is the case shown in *Figure 5*.

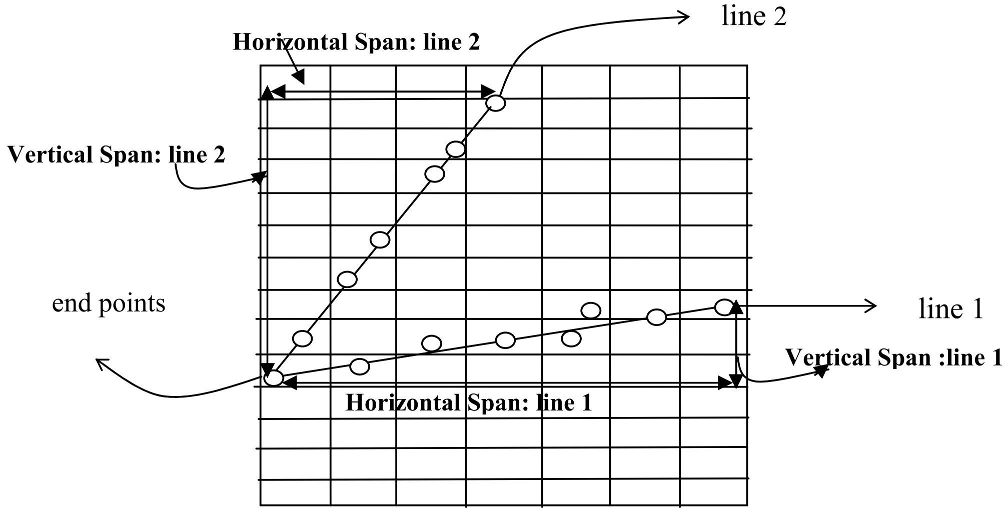


Figure 5: DDA line generation

In *Figure 5*, there are two lines. Line 1 (slope<1) and line 2 (slope>1). Now let us discuss the general mechanism of construction of these two lines with the DDA algorithm. As the slope of the line is a crucial factor in its construction, let us consider the algorithm in two cases depending on the slope of the line whether it is > 1 or < 1.

Case 1: slope (*m*) of line is < 1 (i.e., line 1): In this case to plot the line we have to move the direction of pixel in x by 1 unit every time and then hunt for the pixel value of the y direction which best suits the line and lighten that pixel in order to plot the line.

So, in Case 1 i.e., $0 < m < 1$ where x is to be increased then by 1 unit every time and proper y is approximated.

Case 2: slope (*m*) of line is > 1 (i.e., line 2) if $m > 1$ *i.e.*, case of line 2, then the most appropriate strategy would be to move towards the y direction by 1 unit every time and determine the pixel in x direction which best suits the line and get that pixel lightened to plot the line.

So, in Case 2, i.e., (infinity) > m > 1 where y is to be increased by 1 unit every time and proper x is approximated.

Assumption: The line generation through DDA is discussed only for the I[st] Quadrant, if the line lies in any other quadrant then apply respective transformation (generally reflection transformation), such that it lies in I[st] Quadrant and then proceed with the algorithm, also make intercept Zero by translational transformation such that (x_i, y_i) resolves to $(x_i, mx_i + c)$ or (x_i, mx_i) and similar simplification occurs for other cases of line generation. The concept and application of transformations is discussed in Block 2 of this course.

Note:

1) If in case 1, we plot the line the other way round i.e., moving in y direction by 1 unit every time and then hunting for x direction pixel which best suits the line. In this case, every time we look for the x pixel, it will provide more than one choice of pixel and thus enhances the defect of the stair case effect in line generation. Additionally, from the *Figure 5*, you may notice that in the other way round strategy for plotting line 1, the vertical span is quite less in comparison to the horizontal span. Thus, a lesser number of pixels are to be made *ON*, and will be available if we increase *Y* in unit step and approximate *X*. But more pixels will be available if we increase *X* in unit steps and approximate *Y* (this choice will also reduce staircase effect distortion in line generation) ($\because$ more motion is to be made along x-axis).

2) Consider a line to be generated from (X_0, Y_0) to (X_1, Y_1). If $(X_1 - X_0 > Y_1 - Y_0)$ and $X_1 - X_0 > 0$ then slope (m) of line is < 1 hence case 1 for line generation is applicable otherwise case 2, i.e., If $(X_1 - X_0 < Y_1 - Y_0)$ and $X_1 - X_0 > 0$ then slope m > 1 and case 2 of line generation is applicable.
   Important: Assume that $X_1 > X_0$ is true else swap (X_0, Y_0) and (X_1, Y_1)

3) When $0 < m < 1$: increment *X* in unit steps and approximate *Y*
   - Unit increment should be iterative $\Rightarrow x_i = (x_{i-1}) + 1$ such that (x_i, y_i) resolves to $(x_i, mx_i + c)$ or (x_i, mx_i) . *It is to be noted that while calculating y_i, if y_i turned out to be a floating number then we round its value to select the approximating pixel. This rounding off feature contributes to the staircase effect.*

   When (infinity) > m > 1 increment Y in unit steps and approximate X, simplify (x_i, y_i) accordingly.

Case 1: slope (m) of line is < 1 (or 0 < m < 1)

Consider a line to be generated from (X_0, Y_0) to (X_1, Y_1) , **Assume that $X_1 > X_0$ is true else swap (X_0, Y_0) and (X_1, Y_1)**. Now, if $(X_1 - X_0 > Y_1 - Y_0)$ that means slope (m) of line is < 1 hence, case 1 for line generation is applicable. Thus, we need to increment *X* in unit steps and approximate *Y*. So, from X_0 to X_1 , x_i is incremented in unit steps in the horizontal direction, now for these unit steps the satisfying value of *Y* can be estimated by the general equation of line $y = mx + c$.

Similarly, for case 2, let us sum up our discussion on DDA algorithm for both cases.

We will examine each case separately.

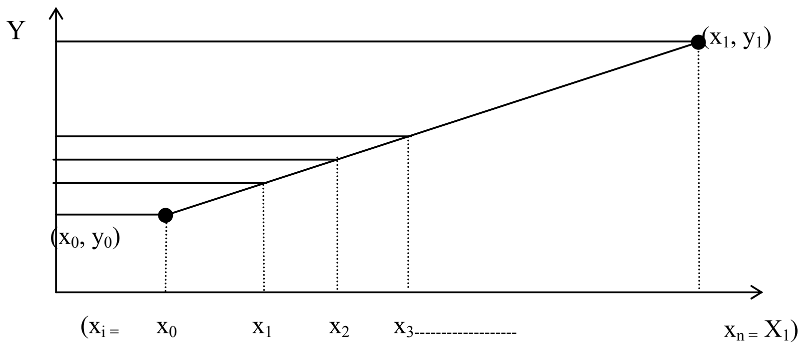


Figure 6: Slope (*m*) of line is < 1 (i.e., line 1)

Sum up:

For instance, in a given line the end points are (X_0, Y_0) and (X_1, Y_1). Using these end points we find the slope ($m = Y_1 - Y_0/ X_1 - X_0$) and check that the value of m lies between 0 and 1 or is > 1. If 0 < m < 1 then case 1 applies, else, case 2 applies.
For case 1, increase x by one Unit every time, for case 2 increase y by one Unit every time and approximate respective values of y and x.

We assume equation of line is $y = mx+c$

At $x = x_i$ we have $y_i = mx_i+c$

Similarly at $x = x_{i+1}$ we have $y_{i+1} = mx_{i+1} +c$

Case 1: Slope (m) of line is 0 < *m*1 (i.e., line 1)

Since x is to be increase by 1 unit each time

$\Rightarrow x_{i+1} = x_i + 1$ ----------- (1)

So by using equation of line $y = mx+c$ we have

$y_{i+1} = m\,(x_i + 1) +c$

$= mx_i +c + m$

$= y_i + m$ ------------ (2)

Equations (1) and (2) imply that to approximate line for case 1 we have to move along x direction by 1 unit to have next value of x and we have to add slope m to initial y value to get next value of y.

Now, using the starting point (x_0, y_0) in the above equations (1) and (2) we go for x_i and y_i (i = 1, 2, ……n) and put colour to the pixel to be lightened.

It is assumed that $X_0 < X_1$ ∴ the algorithm goes like:

```
            x ← X0
            y ← Y0
            m ← (Y1 – Y0)/ (X1 – X0)
      while (x < = X1) do
{     put-pixel (x, round (y), color)
      (new x-value) x ←     (old x-value) x + 1
      (new y-axis)  y ←     (old y-value) y + m
}
```

Sample execution of algorithm case 1:

at (x_0, y_0) : put-pixel (x_0, y_0, colour)

$x_1 = x_0 + 1$; $y_1 = y_0 + m$

at (x_1, y_1) = put pixel $(x_1, y_1$, colour)

similarly, $x_2 = x_1 + 1$; $y_2 = y_1 + m$
at (x_2, y_2) : put pixel $(x_2, y_2$, colour) and so on.

Case 2: slope (*m*) of line is > 1 (i.e., line 2): Same as case 1 but, this time, the y component is increased by one unit each time and appropriate *x* component is to be selected. To do the task of appropriate selection of *x* component we use the equation of Line: $y = mx+c$.

Unit increment should be iterative $\Rightarrow y_{i+1} = y_i + 1$; for this y_{i+1} we find corresponding x_{i+1} by using equation of line $y = mx + c$ and hence get next points (x_{i+1}, y_{i+1}).

$\Rightarrow y_{i+1} - y_i = m\,(x_{i+1} - x_i)$ ----------- (3)

as *y* is to be increase by unit steps
$\therefore y_{i+1} - y_i = 1$ ----------- (4)

using equation (4) in (3) we get
$\Rightarrow 1 = m\,(x_{i+1} - x_i)$ ----------- (5)

rearranging (5) we get

$\Rightarrow 1/m = (x_{i+1} - x_i)$

$$\Rightarrow \boxed{x_{i+1} = x_i + \frac{1}{m}}$$

So, procedure as a whole for Case 2 is summed up as Algorithm case 2:

Assuming $Y_0 < Y_1$ the algorithm goes like

Algorithm for case 2:

```
        x ← X0;
        y ← Y0;
        m ← (Y1 – Y0)/ (X1 – X0);
        m1 ← 1/m;
    while (y < Y1) do
{
    put-pixel (round (x), y, colour)
    y ← y + 1;
    x ← x + m1;
X
}
```

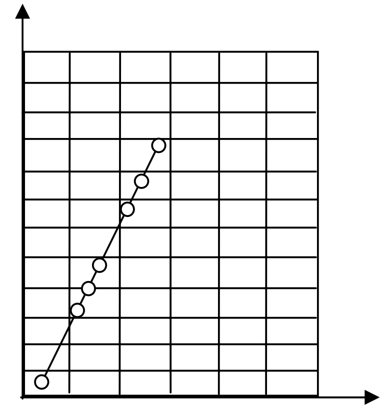


Figure 7: Slope (*m*) of line is > 1

Example 1: Draw line segment from point (2, 4) to (9, 9) using DDA algorithm.

Solution: We know general equation of line is given by

$y = mx+c$ where $m =(y_1 – y_0/(x_1 – x_0)$

given $(x_0, y_0) \rightarrow (2, 4)$; $(x_1, y_1) \rightarrow (9, 9)$

$\Rightarrow m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{9-4}{9-2} = \frac{5}{7}$ i.e., $0 < m < 1$

$C = y_1 - mx_1 = 9 - \frac{5}{7} \times 9 = \frac{63-45}{7} = \frac{18}{7}$

So, by equation of line $(y = mx + c)$ we have

$$y = \frac{5}{7}x + \frac{18}{7}$$

DDA Algorithm Two case:	
Case 1: $m < 1$	$x_{i+1} = x_i + 1$
	$y_{i+1} = y_i + m$
Case 2: $m > 1$	$x_{i+1} = x_i + (1/m)$
	$y_{i+1} = y_i + 1$

As $0 < m < 1$ so according to DDA algorithm case 1

$x_{i+1} = x_i + 1$ $\quad y_{i+1} = y_i + m$

given $(x_0, y_0) = (2, 4)$

1) $x_1 = x_0 + 1 = 3$

$y_1 = y_0 + m = 4 + \frac{5}{7} = \frac{28+5}{7} = \frac{33}{7} = 4\frac{5}{7}$

put pixel (x_0, round y, colour)

i.e., put on (3, 5)

2) $x_2 = x_1 + 1 = 3 + 1 = 4$

$y_2 = y_1 + m = (33/7) + \frac{5}{7} = 38/7 = 5\frac{5}{7}$

put on (4, 5)

Similarly go on till (9, 9) is reached.

☞ Check Your Progress 1

1) What do you mean by the term graphic primitives?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

2) What is the Staircase effect?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

3) What is the reason behind the Staircase effect?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

4) How does the resolution of a system affect graphic display?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

5) Modify the DDA algorithm for negative sloped lines ; discuss both the cases i.e., slope > 1 and 0 < slope < 1.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

6) Using DDA algorithm draw line segments from point (1,1) to (9,7).

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2.4.2 Bresenham Line Generation Algorithm

Bresenham algorithm is accurate and efficient raster line generation algorithm. This algorithm scan converts lines using only incremental integer calculations and these calculations can also be adopted to display circles and other curves.

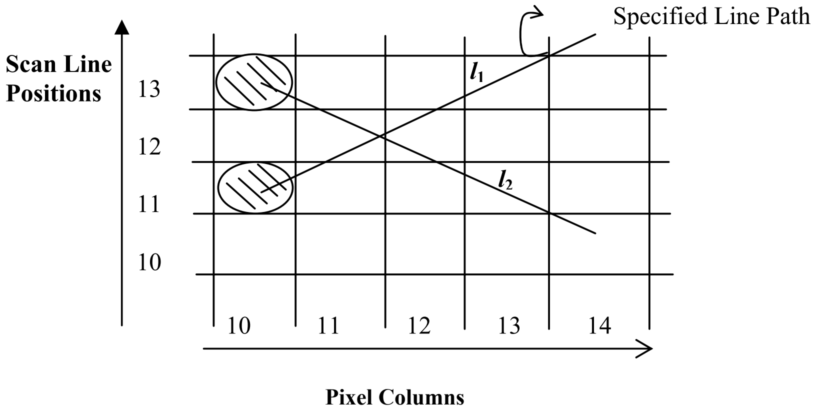


Figure 8: Bresenham line generation

Sampling at Unit *x* distance in *Figure 8*, we need to decide which of the two possible pixel position is closer to the line path at each sample step.

In l_1 : we need to decide that at next sample position whether to plot the pixel at position (11, 11) or at (11, 12).

Similarly, **In l_2**: the next pixel has to be (11, 13) or (11, 12) or what choice of pixel is to be made to draw a line is given by Bresenham, by testing the sign of the integer parameter whose value is proportional to the difference between the separation of the two pixel positions from actual line path. In this section, we will discuss the Bresenham line drawing algorithm for + ve slope ($0 < m < 1$). If the slope is negative then, use reflection transformation to transform the line segment with negative slope to line segment with positive slope. Now, let us discuss the generation of line again in two situations as in the case of DDA line generation.

Case 1: Scan Conversion of Line with the slope ($0 < m < 1$)

Here the pixel positions along the line path are determined by sampling at Unit x intervals *i.e.,* starting from the initial point. (x_0, y_0) of a given line we step to each successive column. (*i.e.*, x-position) and plot the pixel whose scanline y value is closest to the line path. Assume we proceed in this fashion up to the kth step. The process is shown in *Figure 9*. Assuming we have determined the pixel at (x_k, y_k). We need to decide which pixel is to be plotted in column $x_{k+1.}$ Our choices are either (x_{k+1}, y_k) or (x_{k+1}, y_{k+1}).

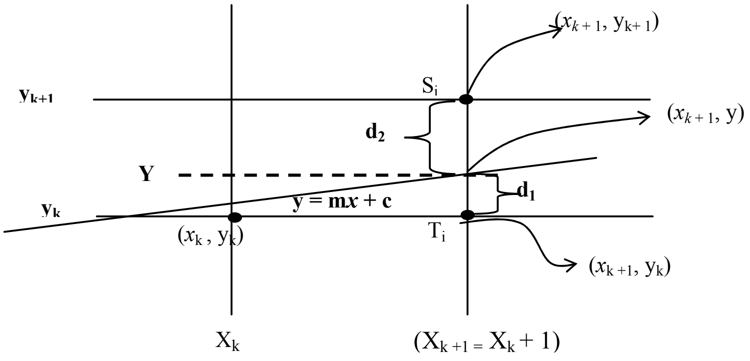


Figure 9: Scan conversion 0 < m < 1

At sampling position X_{k+1} the vertical pixel (or scan line) separation from mathematical line ($y = mx + c$) is say d_1 and d_2.

Now, the y coordinate on the mathematical line at pixel column position X_{k+1} is:

$y = m\,(x_{k+1}) + c$ ---------------------(1)

by *Figure 9*:

$d_1 = y - y_k$ ---------------------(2)

$= m\,(x_{k+1}) + c - y_k$ ---------------------(3) (using (1))

similarly, $d_2 = (y_{k+1}) - y = y_{k+}\,1 - m\,(x_{k+1}) - c$ ---------------------(4)

using (3) and (4) we find $d_1 - d_2$

$d_1 - d_2 = [m\,(x_{k+}\,1) + c - y_k] - [y_k + 1 - m\,(x_{k+}\,1) - c]$

$= mx_k + m + c - y_k - y_k - 1 + mx_k + m + c$

$= 2m\,(x_k + 1) - 2y_k + 2c - 1$ ---------------------(5)

Say, decision parameter p for k[th] step i.e., p_k is given by

$$p_k = \Delta x(d_1 - d_2) \quad \text{---------------------(6)}$$

Now, a decision parameter p_k for the kth step in the line algorithm can be obtained by rearranging (5) such that it involves only integer calculations. To accomplish this substitute $m = \Delta y/\Delta x$ where, Δy and $\Delta x \Rightarrow$ vertical and horizontal separations of the end point positions.

$p_k = \Delta x\,(d_1 - d_2) = \Delta\,x\,[2m\,(x_k + 1) - 2y_k + 2c - 1]$

$= \Delta x\,[2(\Delta y/\Delta x)\;(x_k + 1) - 2y_k + 2c - 1]$

$= 2\,\Delta y\,(x_k + 1) - 2\,\Delta x y_k + 2\,\Delta x c - \Delta x$

$= 2\,\Delta y\,x_k - 2\,\Delta x\,y_k + [2\,\Delta y + \Delta x\,(2c - 1)]$

$p_k = 2\,\Delta y\,x_k - 2\Delta x y_k + b$ -------------------- (7)

where b is constant with value $b = 2\Delta y + \Delta x\,(2c - 1)$ ---------------------(8)

Note: *sign of* p_k is same as sign of $d_1 – d_2$ since it is assumed that $\Delta x > 0$
[As in *Figure 9*, $d_1 < d_2$ *i.e.* $(d_1 – d_2)$ is –ve i.e., p_k is –ve so pixel T_i is more appropriate choice otherwise pixel S_i is the appropriate choice.
i.e., (1) if $p_k < 0$ choose T_i , so next pixel choice after (x_k, y_k) is (x_{k+1}, y_k)
else (2) if $p_k > 0$ choose S_i , so next pixel choice after (x_k, y_k) is (x_{k+1}, y_{k+1}).

At step $k + 1$ the decision parameter is evaluated by writing (7) as:

$$p_{k+1} = 2\Delta y\, x_{k+1} – 2\Delta x\, y_{k+1} + b \qquad \text{---------------------(9)}$$

Subtracting (7) from (9) we get

$$p_{k+1} – p_k = 2\Delta y(\underbrace{x_{k+1} – x_k}_{1}) – 2\Delta x\, (\underbrace{y_{k+1} – y_k}_{\textbf{0 or 1}}) = 2\Delta y – 2\Delta x\, (y_{k+1} – y_k)$$

depending on sign of p_k $\begin{bmatrix} p_k < 0 & y_{k+1} = y_k \\ p_k > 0 & y_{k+1} = y_k + 1 \end{bmatrix}$

$$\boxed{} \qquad \text{---------------------(10)}$$

This recursive calculation of decision parameter is preformed at each integer position, beginning with the left coordinate end point of line.

This first parameter p_0 is determined by using equation (7), and (8) at the starting pixel position (x_0, y_0) with m evaluated as $\Delta y / \Delta x$ (*i.e.,* intercept $c = 0$)

$$p_0 = 0 – 0 + b = 2\Delta y + \Delta x\, (2 * 0 – 1) = 2\Delta y – \Delta x$$

$$p_0 = 2\Delta y – \Delta x \qquad \text{-----------------------(10 A)}$$

Summary:

(Bresenham line drawing algorithm for +ve slope ($0 < m < 1$).

- If slope is negative then use reflection transformation to transform the line segment with negative slope to line segment with a positive slope.

- Calculate constants $2\Delta y$, $2\Delta y – 2\Delta x$, Δy, Δx at once for each line to be scan converted, so the arithmetic involves only integer addition/subtraction of these constants.

Remark : The algorithm will be exactly the same if we assume $| m | < 1$

- **Algorithm | *m* | < *1*:**
  (a) Input two line end points and store left end point in (x_0, y_0)
  (b) Load (x_0, y_0) on frame buffer *i.e.,* plot the first point.
  (c) Calculate Δx, Δy, $2\Delta y$, $2\Delta y – 2\Delta x$ and obtain the starting value of decision parameter as $p_0 = 2\Delta y – \Delta x$
  (d) At each x_k along the line, starting at $k = 0$, perform following test:

  if $p_k < 0$, the next plot is (x_{k+1}, y_k) and $p_{k+1} = p_k + 2\Delta y$
  else next plot is (x_{k+1}, y_{k+1}) and $p_{k+1} = p_k + 2(\Delta y – \Delta x)$

  (e) Repeat step (D) Δx times.

Bresenham Line Generation Algorithm (| m | < 1)

$\Delta x \leftarrow x_1 – x_0$
$\Delta y \leftarrow y_1 – y_0$
$p_0 \leftarrow 2\Delta y – \Delta x$

while $(x_0 <= x_1)$ do
{puton (x_0, y_0)
if (pi > 0) then
{$x_0 \leftarrow x_0 + 1$;
$y_0 \leftarrow y_0 + 1$;
$p_{i+1} \leftarrow pi + 2\ (\Delta y - \Delta x)$;

}
if (pi < 0) then
{$x_0 \leftarrow x_0 + 1$
$y_0 \leftarrow y_0$
$p_{i+1} \leftarrow pi + 2\ \Delta y$
}
}

Note:

- Bresenhams algorithm is generalised to lines with arbitrary slopes by considering the symmetry between the various octants and quadrants of the coordinate system.
- For line with +ve slope (m) such that m > 1, then we interchange the roles of *x* and *y* direction i.e., we step along *y* directions in unit steps and calculate successive *x* values nearest the line path.
- for –ve slopes the procedures are similar except that now, one coordinate decreases as the other increases.

Example 2: Draw line segment joining (20, 10) and (25, 14) using Bresenham line generation algorithm.

Solution: $(x_0, y_0) \rightarrow (20, 10)$; $(x_1, y_1) \rightarrow (25, 14)$

$$m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{14-10}{25-20} = \frac{4}{5} < 1$$

As, $m = \frac{\Delta y}{\Delta x} = \frac{4}{5} \Rightarrow \Delta y = 4$

$\Delta x = 5$

$\rightarrow$ plot point (20, 10)

$p_i = 2\Delta y - \Delta x$

$i = 1$: $p_i = 2 * 4 - 5 = 3$

as $p_1 > 0$ so $x_0 \leftarrow 21$; $y_0 \leftarrow 11$

now plot (21,11)

$i = 2$ as $p_1 > 0$

$\therefore p_2 = p_1 + 2(\Delta y - \Delta x)$

$= 3 + 2\ (4 - 5) = 3 - 2 = 1$

$p_2 > 0$ so $x_0 \leftarrow 22$; $y_0 \leftarrow 12$

plot (22,12)

$i = 3$ as $p_2 > 0$

$\therefore p_3 = p_2 + 2\ (\Delta y - \Delta x) = 1 + 2\ (4 - 5) = -1$

$p_3 < 0$ $\therefore x_0 \leftarrow 23$

$y_0 \leftarrow 12$

plot (23, 12)

$i = 4$ as $p_3 < 0$

$\therefore p_4 = p_3 + 2\Delta y$

$= -1 + 2 * 4 = 7$

$\therefore x_0 \leftarrow 24;\ y_0 \leftarrow 13$
plot (24, 13)
$i = 5$ as $p_4 > 0$

$\therefore p_5 = p_4 + 2\,(\Delta y - \Delta x)$
$= 7 + 2\,(4 - 5) = 5$
$x_0 \leftarrow 25;\ y_0 \leftarrow 14$

plot (25, 14)

{for i = 6, x_0 will be > x_i so algorithm terminates

Example 3: Illustrate the Bresenham line generation algorithm by digitzing the line with end points (20, 10) and (30,18)

Solution: $m = \dfrac{y_2 - y_1}{x_2 - x_1} = \dfrac{\Delta y}{\Delta x} = \dfrac{18-10}{30-20} = \dfrac{8}{10} = 0.8$ -------------------(1)

$\Rightarrow \Delta y = 8$ and $\Delta x = 10$ -------------------(2)

value of initial decision parameter (p_0) = $2\Delta y - \Delta x = 2 * 8 - 10 = 6$ -----------------(3)

value of increments for calculating successive decision parameters are:

$2\Delta y = 2 * 8 = 16;$ -------------------(4)

$2\Delta y - 2\Delta x = 2 * 8 - 2 * 10 = -4$ -------------------(5)

plot initial point (x_0, y_0) = (20, 10) in frame buffer now determine successive pixel positions along line path from decision parameters value (20, 10).

k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)
1	2	(22, 12)
2	– 2	(23, 12)
3	14	(24, 13)
4	10	(25, 14)
5	6	(26, 15)
6	2	(27, 16)
7	– 2	(28, 16)
8	14	(29, 17)
9	10	(30, 18)

← [use step (d) of algorithm Δx times]

If $p_k > 0$ then increase both X and Y and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$

If $p_k < 0$ then increase X and not Y and $p_{k+1} = p_k + 2\Delta y$

☞ Check Your Progress 2

1) Compare Bresenham line generation with DDA line generation.

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

……………………………………………………………………………………………

2) Illustrate the Bresenham line generation algorithm by digitising the line with end points (15, 5) and (25,13).

………………………………………………………………………………………

………………………………………………………………………………………

2.5 CIRCLE-GENERATION ALGORITHMS

Circle is one of the basic graphic component, so in order to understand its generation, let us go through its properties first:

2.5.1 Properties of Circles

In spite of using the Bresenham circle generation (i.e., incremental approval on basis of decision parameters) we could use basic properties of circle for its generation.

These ways are discussed below:

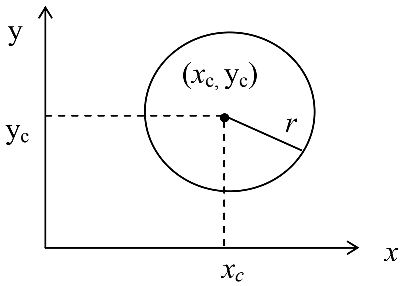


Figure 8: Property of circle

Generation on the basis of Cartesian Coordinates:

A circle is defined as the set of points or locus of all the points, which exist at a given distance r from center (x_c, y_c). The distance relationship could be expressed by using Pythagonous theorem in Cartesian coordinates as

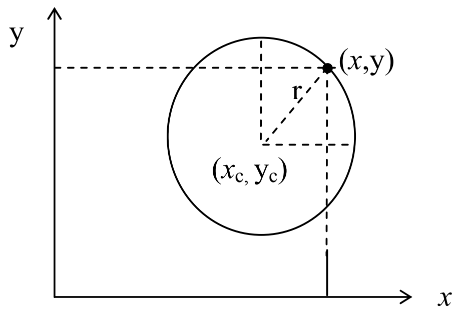


Figure 9: Circle generation (cartesian coordinate system)

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \qquad \text{---------------------(1)}$$

Now, using (1) we can calculate points on the circumference by stepping along x-axis from $x_c – r$ to $x_c + r$ and calculating respective y values for each position.

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2} \qquad \text{---------------------(2)}$$

Generation on basis of polar coordinates (r and θ)

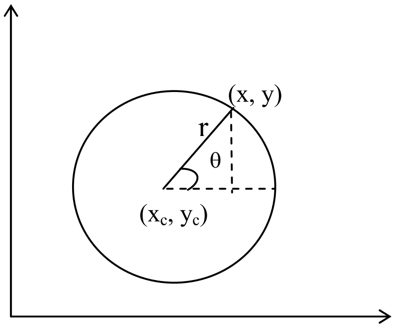


Figure 10: Circle generation (polar coordinate system)

Expressing the circle equation in parametric polar form yields the following equations

$$x = x_c + r\cos\theta$$

$$y = y_c + r\sin\theta \qquad \text{-------------------------(3)}$$

Using a fixed angular step size we can plot a circle with equally spaced points along the circumference.

Note:

- *Generation of circle with the help of the two ways mentioned is not a good choice ∵ both require a lot of computation equation (2) requires square root and multiplication calculations where as equation (3) requires trigonometric and multiplication calculations. A more efficient approach is based on incremental calculations of decision parameter. One such approach is Bresenham circle generation. (mid point circle algorithm*).

- This *Bresenham circle generation (mid point circle algorithm*) is similar to line generation. Sample pixels at Unit x intervals and determine the closest pixel to the specified circle path at each step.

- For a given radius and center position (x, y,) we first setup our algorithm to calculate pixel position around the path of circle centered at coordinate origin (0, 0) *i.e.,* we translate $(x_c, y_c) \to (0, 0)$ and after the generation we do inverse translation $(0, 0) \to (x_c, y_c)$ hence each calculated position (x, y) of circumference is moved to its proper screen position by adding x_c to x and y_c to y.

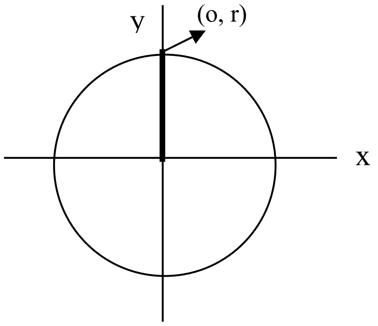


Figure 11: Circle generation (initialisation)

- In *Bresenham circle generation (mid point circle algorithm)* we calculate points in an octant/quadrant, and then by using symmetry we find other respective points on the circumference.

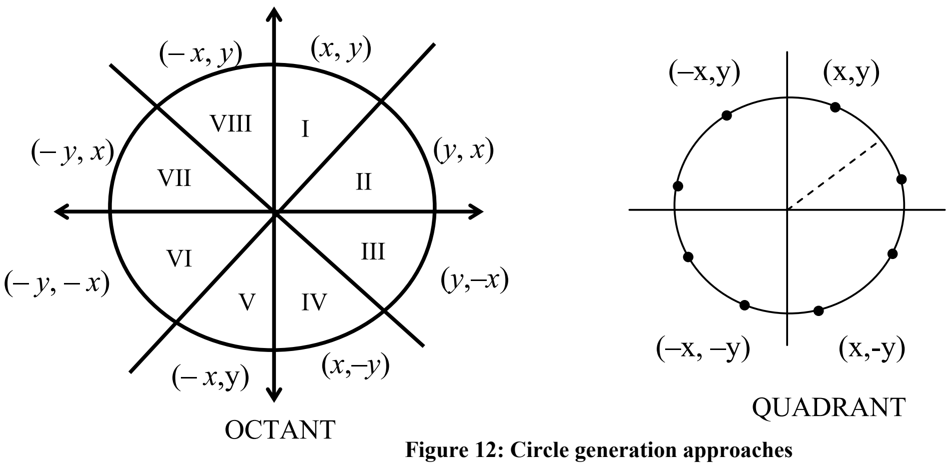


Figure 12: Circle generation approaches

2.5.2 Mid Point Circle Generation Algorithm

We know that equation of circle is $x^2 + y^2 = r^2$. By rearranging this equation, we can have the function, to be considered for generation of circle as $f_c (x, y)$.

$$f_c (x, y) = x^2 + y^2 - r^2 \qquad \text{--------------(1)}$$

The position of any point (x, y) can be analysed by using the sign of $f_c (x, y)$ i.e.,

$$\text{decision parameter} \rightarrow f_c (x, y) = \begin{cases} < 0 \text{ if } (x, y) \text{ is inside circle boundary} \\ = 0 \text{ if } (x, y) \text{ is on the circle boundary} \\ > 0 \text{ if } (x, y) \text{ is outside circle boundary} \end{cases} \qquad \text{---------(2)}$$

i.e., we need to test the sign of $f_c (x, y)$ are performed for mid point between pixels near the circle path at each sampling step.

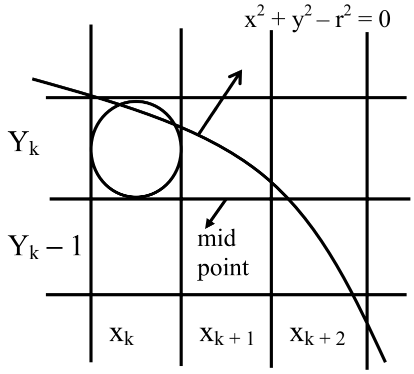


Figure 13: Mid point between Candidate Pixel at sampling Position x_{k+1} along Circular Path

Figure 13 shows the mid point of two candidate pixel (x_{k+1}, y_k) and (x_{k+1}, y_{k-1}) at sampling position $x_k + 1$.

Assuming we have just plotted the (x_k, y_k) pixel, now, we need to determine the next pixel to be plotted at (x_{k+1}, y_k) or (x_{k+1}, y_{k-1}). In order to decide the pixel on the circles path, we would need to evaluate our decision parameter p_k at mid point between these two pixels i.e.,

$$p_k = f_c (x_{k+1}, y_k - \frac{1}{2}) = (x_{k+1})^2 + (y_k - \frac{1}{2})^2 - r^2 \qquad \text{---------------------(3)}$$

(using (1)

If $p_k < 0$ then $\Rightarrow$ midpoint lies inside the circle and pixel on scan line y_k is closer to circle boundary else mid point is outside or on the circle boundary and we select pixel on scan line $y_k - 1$ successive decision parameters are obtained by using incremental calculations.

The recursive way of deciding parameters by evaluating the circle function at sampling positions $x_{k+1} + 1 = (x_k + 1) + 1 = x_k + 2$

$$p_{k+1} = f_c\left(x_{k+1} + 1,\ y_{k+1} - \frac{1}{2}\right) = \left[(x_k + 1) + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2\right] \quad \text{--------(4)}$$

subtract equation (3) from (4) we get

$$p_{k+1} - p_k = \left\{[(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2\right\} - \left\{ (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2\right\}$$

$$p_{k+1} - p_k = [(x_k + 1)^2 + 1 + 2(x_k + 1).\ 1] + \left[y^2_{k+1} + \frac{1}{4} - \cancel{2}.\frac{1}{\cancel{2}}.\ y_{k+1}\right] - \cancel{r^2} - (x_k + 1)^2$$

$$- \left[y_k^2 + \frac{1}{4} - \cancel{2}.\ \frac{1}{\cancel{2}}.\ y_k\right] + \cancel{r^2}$$

$$= \cancel{(x_k + 1)^2} + 1 + 2\ (x_k + 1) + y^2_{k+1} + \cancel{\tfrac{1}{4}} - y_{k+1} - \cancel{(x_k + 1)^2} - y_k^2 - \cancel{\tfrac{1}{4}} + y_k$$

$$p_{k+1} = p_k + 2\ (x_k + 1) + (y^2_{k+1} - y_k^2) - (y_{k+1} - y_k) + 1 \quad \text{--------------(5)}$$

Here, y_{k+1} could be y_k or y_{k-1} depending on sign of p_k

$$\left[\begin{array}{l} \text{so, increments for obtaining } p_{k+1} \text{ are : -} \\ \text{either } 2x_{k+1} + 1 (\text{if } p_k < 0)\ (i.e., y_{k+1} = y_k) \\ or\ 2x_{k+1} + 1 - 2y_{k-1} (\text{if } p_k > 0)\ (i.e., y_{k+1} = y_k - 1)\ So (2y_{k+1} = 2(y_k - 1) = 2y_k - 2 \end{array}\right.$$

How do these increments occurs?

- $y_{k+1} = y_k$: use it in (5) we get
  $p_{k+1} = p_k + 2\ (x_k + 1) + 0 - 0 + 1 = p_k + (2x_{k+1} + 1)$

$\Rightarrow$ ----------------(6)

- $y_{k+1} = y_{k-1}$ use it in (5) we get
  $p_{k+1} = p_k + 2x_{k+1} + (y_{k+1}^{\ 2} - y_k^2) - (y_{k+1} - y_k) + 1$
  $= p_k + 2x_{k+1} + \underline{(y_{k+1} - y_k)}\ \underline{[(y_{k+1} + y_k) - 1]} + 1$

$$\boxed{p_{k+1} = p_k + (2x_{k+1} - 2y_{k-1} + 1)} \quad \text{------------------(7)}$$

Also,

$2x_{k+1} = 2\ (x_k + 1) = 2x_k + 2$ ------------------ (8)

$2y_{k+1} \rightarrow 2y_{k-1} = 2\ (y_k - 1) = 2y_k - 2$ -------------------(9)

Note:

- At starting position $(0, r)$ the terms in (8) and (9) have value 0 and $2r$ respectively. Each successive value is obtained by adding 2 to the previous value of $2x$ and subtracting 2 from the previous value of $2y$.

- The initial decision parameter is obtained by evaluating the circle function at start position $(x_0, y_0) = (0, r)$

  i.e., $p_0 = f_c\left(1, r - \frac{1}{2}\right) = 1 + \left(r - \frac{1}{2}\right)^2 - r^2 = \frac{5}{4} - r$ (using 1)

  If radius r is specified as an integer then we can round p_0 to

  $$\boxed{p_0 = 1 - r} \quad \text{(for } r \text{ as an integer).}$$

Midpoint Circle Algorithm

(a) Input radius r and circle, center (x_c, y_c) and obtain the first point on the circumference of the circle centered on origin as

$$(x_0, y_0) = (0, r)$$

(b) Calculate initial value of decision parameter as

$$p_0 = \frac{5}{4} - r \sim 1 - r$$

(c) At each x_k position starting at $k = 0$ perform following test.

1) If $p_k < 0$ then next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and $p_{k+1} = p_k + 2x_{k+1} + 1$
2) Else the next point along circle is $(x_{k+1}, y_k - 1)$ and
   $p_{k+1} = p_k + 2x_{k+1} - 2y_{k-1}$
   where $2x_{k+1} = 2(x_k + 1) = 2x_k + 2$ and $2y_{k-1} = 2(y_k - 1) = 2y_k - 2$

(d) Determine symmetry points in the other seven octants.

(e) Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot coordinate values $x = x + x_c$, $y = y + y_c$

(f) Repeat step (c) through (e) until $x \geq y$.

Example 4: Given a circle radius $r = 10$ determine positions along the circle octants in 1st Quadrant from $x = 0$ to $x = y$.

Solution: An initial decision parameter $p_0 = 1 - r = 1 - 10 = -9$
For circle centered on coordinate origin the initial point $(x_0, y_0) = (0, 10)$ and initial increment for calculating decision parameter are:

$$2x_0 = 0, 2y_0 = 20$$

Using mid point algorithm point are:

k	p_k	(x_{k+1}, y_{k-1})	$2x_{k+1}$	$2y_{k-1}$
0	– 9	(1, 10)	2	20
1	– 6	(2, 10)	4	20
2	– 1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	– 3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7, 7)	14	14

2.6 POLYGON FILLING ALGORITHM

In many graphics displays, it becomes necessary to distinguish between various regions by filling them with different colour or at least by different shades of gray. There are various methods of filling a closed region with a specified colour or

equivalent gray scale. The most general classification appears to be through following algorithms:

1) Scan Line polygon fill algorithm.
2) Seed fill algorithm./Flood fill algorithm which are further classified as:
   (a) Boundary fill algorithm
   (b) Interior fill algorithm

We restrict our discussion to scan line polygon fill algorithm, although we will discuss the others in brief at the end of this section.

Scan Line Polygon Fill Algorithm

In this, the information for a solid body is stored in the frame buffer and using that information all pixels i.e., of both boundary and interior region are identified and, are hence plotted.

Here we are going to do scan conversion of solid areas where the region is bounded by polygonal lines. Pixels are identified for the interior of the polygonal region, and are then filled plotted with the predefined colour.

Let us discuss the algorithm briefly, then we will go into details on the same.

This algorithm checks and alters the attributes (i.e., characteristics and parameters) of the pixels along the current raster scan line. As soon as it crosses over from the outside to the inside of a boundary of the specified polygon it starts resetting the colour (or gray) attribute. In effect filling the region along that scan line. It changes back to the original attribute when it crosses the boundary again. *Figure 14* shows variations of this basic idea.

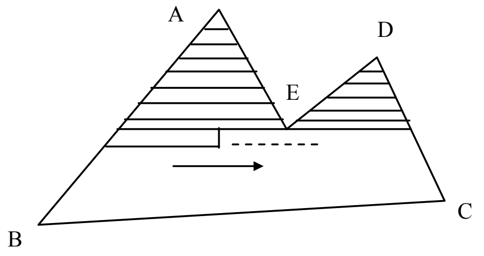


Figure 14: Concept of scan line polygon filling

So as to understand Scan Line Polygon Fill Algorithm in detail consider *Figure 15*:

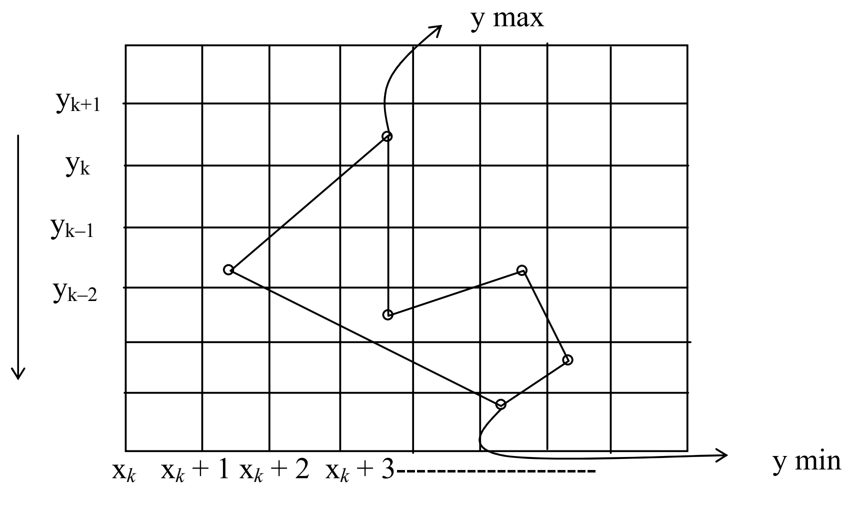


Figure 15: Scan line polygon filling

Here, in *Figure 15,*

- y_k, y_{k-1},….. → are scan lines from top to bottom (value of y at top or scan line at top has maximum value and the scan line at bottom has minimum y value).
- x_k, $x_k + 1$…..→are consecutive pixels on a scan line *i.e.,* (x_k, y_k), (x_{k+1}, y_k),………..⇒ a sea of pixels for scan lines y_k, y_{k-1}, y_{k-2}, ….., y_i and for the chosen scan line we find pixels satisfying the condition as mentioned above.
- For a given scan line, let us assume that it intersects edges of the given polygonal region at the points *P1, P2, ..., Pk.* Sort these points *P1, P2, ..., Pk* in terms of the *X*-coordinates in increasing order.

Now, for a chosen scan line there are three cases:

(a) Scan line passes through the edges in between shown by point a in *Figure 16.*
(b) Scan line passes through the vertex of the polygon whose neighbouring vertices lie completely on one side of the scan line (point b, point c in *Figure 16*).
(c) Scan line passes through some bottom vertex of polygon whose neighbouring vertices lie on both sides of the scan line.

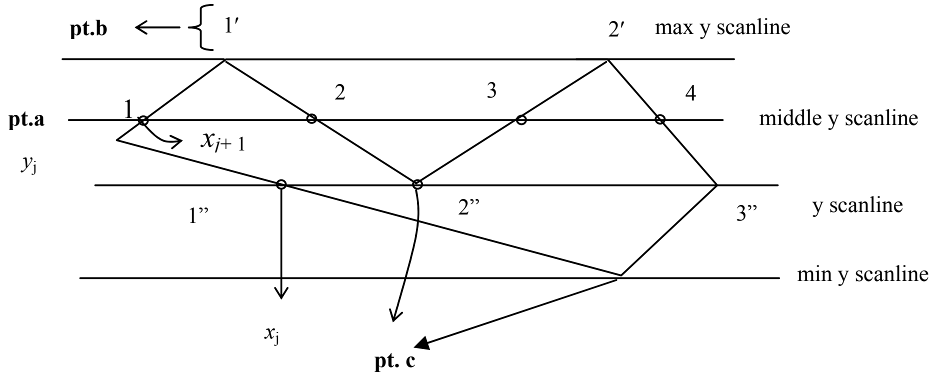


Figure 16: Cases for scan line polygon filling

In case a, *i.e.,* when a scan line intersects an edge at a point but not a vertex point of the edge (as in case of point 1, 2, 3, 4) then that point of intersection is considered as a single point and it is taken as ON/OFF point depending on the requirement. Therefore, in this case the intersection points are taken as 1234. Colour of the pixels between the intersection points 1 and 2 is replaced with the filling colour but the colour of the pixels between the points 2 and 3 are left unchanged. Again the colour of the pixels between 3 and 4 are changed by the filling colour.

In case b and c, case *c* i.e., when a scan line intersects an edge E1 at a vertex V1 i.e., a vertex point of the edge E1 whose *y* coordinate is greater (or less) than the *y* coordinate values of the other two vertex points V2 and V3 where, for example, edge E1 has end vertex points V1, V2 and edge E2 having end vertex points V1, V3. That is the vertices V2 and V3 will lie either completely above V1 or both V2 and V3 will lie completely below the vertex V1. In this case, the point of intersection, i.e., E1, will be counted two times. For example, the vertices 1′ and 2′ are such that their y-coordinate values are grater than the y-coordinate values of their neighbouring vertices and therefore they are counted twice.1′1′ 2′2′ i.e. the colour of the pixels between 1′, 1′ is replaced with the filling colour but the colour of the pixels between 1′ , 2′ is left unchanged. Again the colour of the pixels between 2′, 2′ is changed.

Assume that the scan line passes through a vertex point V1 of the polygonal region having neighbouring vertices V0 and V2., i.e. let E1 and E2 be two edges of the polygon so that V0, V1 and V1, V2 be respectively their end vertices. Suppose we assume that the vertex V1 is such that the y-coordinate for the neighbouring vertex V0 is grater than the y-coordinate for V1 but the y-coordinate for the neighbouring vertex

V2 is less than the y-coordinate for V1. In this case, the intersection vertex point V0 will be taken as a single point and the algorithm will remain the same as above.

The point of intersection is to be considered two times, i.e., 1″ 2″ 2″ 3″. 1″ 2″ ⇒ make pixels ON from 1″ to 2″, 2″ 2″ don't make pixel ON, 2″, 3″⇒ make pixels ON from 2″ to 3″.

Now, the requirements for scanning an image are:

1) Determine the point of intersection of an edge and scan line

2) Sort the points on scan line w.r.t x value. Get the respective pixels ON.

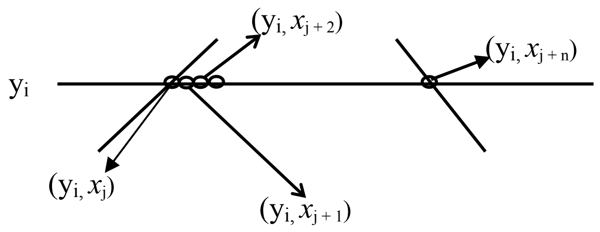


Figure 17: Requirement for image scanning

Sum up: To fill a polygonal image we have to identify all edges and vertices of intersections while moving from scan line y_{max} to y_{min}, for each scan line. Then check for the case and perform the algorithm accordingly.

Recursive way to scan a figure

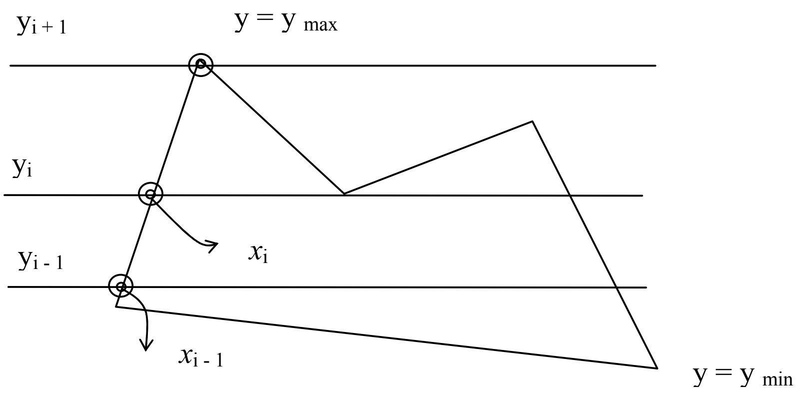


Figure 18: Scan line polygon filling (a recursive approach)

For a scan line $y = y_i$ identify the edges and use the data of the point of intersection (x_i ,y_i.). Say x_0 is point on scan line y_0 and x_1 is point on scan line y_1. Now between (x_0, y_0) and (x_1, y_1) we find slope (this all is to find recursive way to scan a figure).

x_0	y_0	x_1	y_1	$1/m$

$$\because \quad m = y_1 - y_0/x_1 - x_0 = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

$$\Rightarrow m\ (x_{i+1} - x_i) = (y_{i+1} - y_i) \qquad \text{-------------(1)}$$

now y_{i+1} is scan line next to y_i i.e. [ ] -------------(2)

($\because$ moving from top to bottom we are moving from y_{max} to y_{min})

Using (2) in (1) we get

$$m\,(x_{i+1} - x_i) = (y_{i+1} - y_i) = (y_i + 1 - y_i)$$

$$\Rightarrow \quad \boxed{x_{i+1} = x_i - (1/m)}$$

Using y_{i+1} and x_{i+1} expressions we can find consecutive points. x for respective scan lines y_i . If x_i comes out to be real then round the value to integer values.

Note: Although Flood fill algorithms are not part of this block let me briefly describe Flood Fill Algorithms. The algorithm starts with the coordinates of a specified point (known as the "seed" pixel) inside the polygon to be filled, and proceeds outwards from it, testing the adjacent pixels around it in orderly fashion until the spreading wave reaches the boundary in every direction, as suggested in *Figure 19*.

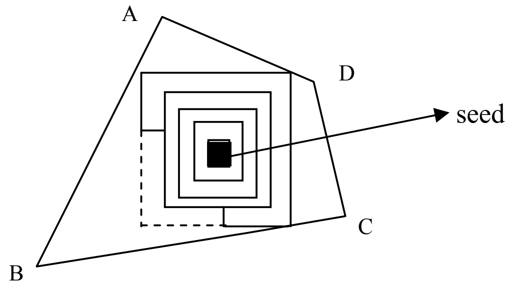


Figure 19: Flood filling method

To economise in storage and memory management, a combination of the two techniques may be used. Starting from an interior seed, first the scan-line through the seed is painted until the left and right boundaries are reached, then the line above and below are similarly covered. The procedure is repeated until the entire region is filled.

Filling may be solid (painted with a single colour), or it may be patterned or tiled (filled with different patterns).

Fill options in standard packages such as MS-Word are grouped generally under the heads: (a) Gradient, variations from light to dark of specified colour in specified directions; (b) Texture; (c) Pattern (i.e., Hatching); and (d) Picture, of the user's choice. Some of these will be discussed later.

☞ Check Your Progress 3

1) Do we need to generate the full circumference of the circle using the algorithm, or can we can generate it in a quadrant or octant only and then use it to produce the rest of the circumference?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) Given a circle radius $r = 5$ determine positions along the circle octants in 1st Quadrant from $x = 0$ to $x = y$?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) Distinguish between Scan line polygon fill and Seed fill or Flood fill algorithm?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2.7 SUMMARY

In this unit, we have discussed the basic graphic primitives that is point, line and circle; we have also discussed both theoretical and practical application of different algorithms related to their generation. At the end of the unit, we have emphasised on different seed fill and flood fill type of polygon filling algorithms. The filling algorithms are quite important as they give privilege to quickly fill colours into the graphics created by you.

2.8 SOLUTIONS/ANSWERS

Check Your Progress 1

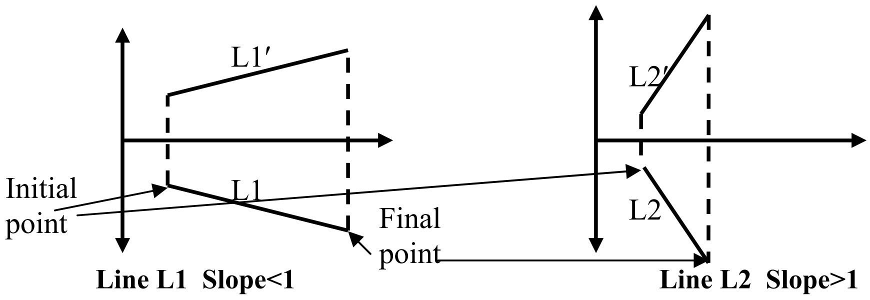
1) Graphic primitives are the fundamental graphic objects which can be united in any number and way to produce a new object or image.

2) Due to low resolution and improper approximation of pixel positions the images are generally distorted and the Zig-Zags (i.e., distortions) produced in the display of straight line or any other graphic primitive is the staircase effect.

3) The approximation involved in the calculation for determination of pixel position involved in the display of lines and other graphic primitives is the actual cause of this effect.

4) In a high resolution system the adjacent pixels are so closely spaced that the approximated line-pixels lie very close to the actual line path and hence the plotted lines appear to be much smoother — almost like straight lines drawn on paper. In a low resolution system, the same approximation technique causes lines to be displayed with a “stairstep apperance” i.e., not smooth.

5)



For the generation of lines with negative slopes:

Slope < 1 : successively increase y and respectively decrease x
Slope > 1 : successively increase x and respectively decrease y

Aliter: This hint you will understand after going through Block 2.

In the shown Figure say L1 and L2 are lines with negative slope with a magnitude of <1 and >1 respectively. Then for the generation of such a line, you may take reflection of the line about the concerned axis and follow the usual algorithm of line generation. After that you may inverse reflection the transformation, and this will provide you the line generation of negative slope.

6) We know that the general equation of the line is given by
$y = mx+c$ where $m =(y_1 – y_0/(x_1 – x_0)$

given $(x_0, y_0) \rightarrow (1, 1)$; $(x_1, y_1) \rightarrow (9, 7)$

$\Rightarrow m = (7-1)/(9-1) =6/8$
$C = y_1 – mx_1 = 7 – (6/8) *9 = 1/4$

So, by equation of line $(y = mx + c)$ we have
$y = (6/8)x+(1/4)$

DDA Algorithm Two case:		
Case 1: $m < 1$	$x_{i+1} = x_i + 1$	
	$y_{i+1} = y_i + m$	
Case 2: $m > 1$	$x_{i+1} = x_i + (1/m)$	
	$y_{i+1} = y_i + 1$	

as m < 1 so according to DDA algorithm case 1
$x_{i+1} = x_i + 1$ $\quad$ $y_{i+1} = y_i + m$
given (x_0, y_0) = (1, 1)

1) $x_1 = x_0 + 1 = 2$
$y_1 = y_0 + m = 1+(6/8) = 7/4$

put pixel (x_0, round y, colour)
i.e., put on (2, 2)

Similarly, go on till (9, 7) is arrived at.

Check Your Progress 2

1) Bresenham line generation algorithm is better than DDA because it has better solution to the concept of approximation of pixel position to be enlightened for the display of any graphic primitive, thus, reduces the staircase effect, for more details refer to 2.4.2

2) Here we are using the Bresenham line generation algorithm by digitising the line with end points (15, 5) and (25,13).

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{13-5}{25-15} = \frac{8}{10} = 0.8 \qquad \text{------------------(1)}$$

$\Rightarrow \Delta y = 8$ and $\Delta x = 10$ -------------------(2)

value of initial decision parameter $(p_0) = 2\Delta y - \Delta x = 2 * 8 - 10 = 6$ ------------(3)

value of increments for calculating successive decision parameters are:

$2\Delta y = 2 * 8 = 16;$ -------------------(4)

$2\Delta y - 2\Delta x = 2 * 8 - 2 * 10 = -4$ -------------------(5)

plot initial point $(x_0, y_0) = (15, 5)$ in the frame buffer now determine successive pixel positions along line path from decision parameters value (15, 5).

k	**p$_k$**	**(x_{k+1}, y_{k+1})**	← **[use step (d) of algorithm Δx time]**
0	6	(16, 6)	
1	2	(17, 7)	
2	– 2	(18, 7)	
3	14	(19, 8)	
4	10	(20, 9)	
5	6	(21, 10)	
6	2	(22, 11)	
7	– 2	(23, 11)	
8	14	(24, 12)	
9	10	(25, 13)	

> If $p_k > 0$ then increase both X and Y and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
>
> If $p_k < 0$ then increase X and not Y and $p_{k+1} = p_k + 2\Delta y$

Check Your Progress 3

1) No, there is no need to generate the full circumference of the circle using the algorithm. We may generate it in a quadrant or octant only and then use it to produce the rest of the circumference by taking reflections along the respective axis.

2) Refer to the discussed example 4 on page 16.

3)

Scan Line Polygon	**Flood Fill Algorithms**
• This algorithm checks and alters the attributes (i.e., characteristics and parameters) of the pixels along the current raster scan line. As soon as it crosses from outside to the inside of a boundary of the specified polygon or other region, it starts resetting the colour (or gray) attribute, in effect filling the region along that scan line. It changes back to the original attribute when it crosses the boundary again.	• Flood Fill Algorithms**.** In these, the algorithm starts with the coordinates of a specified point (known as the "seed") inside the polygon to be filled, and proceeds outwards from it, testing the adjacent pixels around it in orderly fashion, until the spreading wave reaches the boundary in every direction.
• Polygon filling is time consuming.	• Polygon filling is quite quick.

UNIT 3 2-D VIEWING AND CLIPPING

Structure **Page No.**

3.1	Introduction	71
3.2	Objectives	72
3.3	Point Clipping	73
3.4	Line Clipping	73
	3.4.1 Cohen Sutherland Line Clippings	73
	3.4.2 Cyrus-Beck Line Clipping Algorithm	84
3.5	Polygon Clipping	90
	3.5.1 Sutherland-Hodgman Algorithm	91
3.6	Windowing Transformations	93
3.7	Summary	97
3.8	Solutions/Answers	98

3.1 INTRODUCTION

In the earlier two units of this block, we discussed the basic components of computer graphics, i.e., the software and hardware used for graphic systems along with the graphic primitives used to create graphic images. Now, we are in a position to discuss technicalities related to the display of any graphic primitive or image. Let us begin with, the concept of clipping, for both point and line clipping, followed by the concerned algorithms used to perform line clipping. We shall then examine the concept and algorithm related to polygon clipping, and end with a discussion on window to viewport transformations.

Clipping may be described as the procedure that identifies the portions of a picture lie inside the region, and therefore, should be drawn or, outside the specified region, and hence, not to be drawn. The algorithms that perform the job of clipping are called clipping algorithms there are various types, such as:

- Point Clipping
- Line Clipping
- Polygon Clipping
- Text Clipping
- Curve Clipping

Further, there are a wide variety of algorithms that are designed to perform certain types of clipping operations, some of them which will be discussed in unit.

Line Clipping Algorithms:
- Cohen Sutherland Line Clippings
- Cyrus-Beck Line Clipping Algorithm

Polygon or Area Clipping Algorithm
- Sutherland-Hodgman Algorithm

There are various other algorithms such as, Liang – Barsky Line clipping, Weiler-Atherton Polygon Clipping, that are quite efficient in performing the task of clipping images. But, we will restrict our discussion to the clipping algorithms mentioned earlier.
Before going into the details of point clipping, let us look at some basic terminologies used in the field of clipping, such as, window and viewport.

Window may be described as the world coordinate area selected for display.

Viewport may be described as the area on a display device on which the window is mapped.

So, it is the window that specifies what is to be shown or displayed whereas viewport specifies where it is to be shown or displayed.

Specifying these two coordinates, i.e., window and viewport coordinates and then the transformation from window to viewport coordinates is very essential from the point of view of clipping.

Note:

- Assumption: That the window and viewport are rectangular. Then only, by specifying the maximum and the minimum coordinates *i.e.,* (Xw_{max}, Yw_{max}) and (Xw_{min}, Yw_{min}) we can describe the size of the overall window or viewport.
- Window and viewport are not restricted to only rectangular shapes they could be of any other shape (Convex or Concave or both).

For better understanding of the clipping concept refer to *Figure 1*:

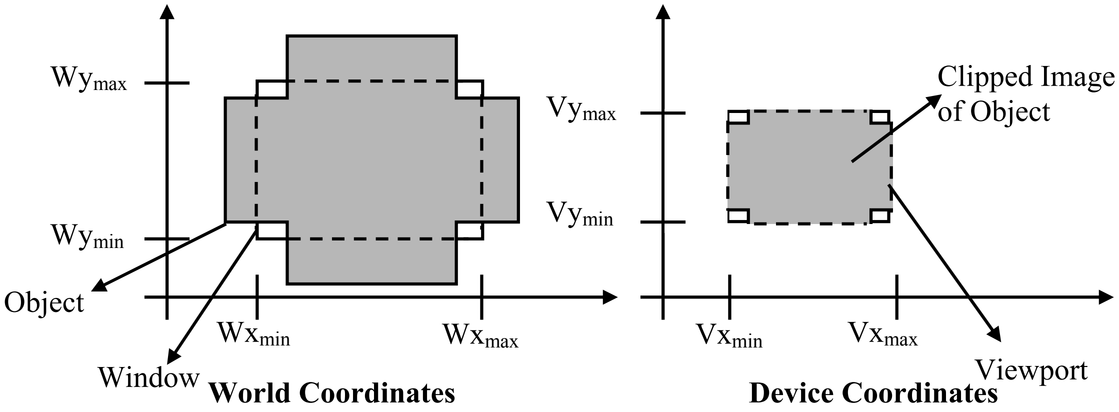


Figure 1: Clipping alongwith Viewing Transformation through rectangular window and viewport

3.2 OBJECTIVES

After going though this unit, you should be able to:

- explain the concept of clipping,
- examine how line clipping is performed,
- understand and implement the algorithms that work behind the concept of line clipping,
- explain how polygon clipping is performed,
- understand and implement the algorithms that work behind the concept of polygon clipping, and
- describe window to viewport transformation.

3.3 POINT CLIPPING

Point clipping is the technique related to proper display of points in the scene, although, this type of clipping is used less frequently in comparison to other types, i.e., line and polygon clipping. But, in some situations, e.g., the scenes which involve particle movements such as explosion, dust etc., it is quite useful. For the sake of simplicity, let us assume that the clip window is rectangular in shape. So, the minimum and maximum coordinate value, i.e., (Xw_{max}, Yw_{max}) and (Xw_{min}, Yw_{min})

are sufficient to specify window size, and any point (X,Y), which can be shown or displayed should satisfy the following inequalities. Otherwise, the point will not be visible. Thus, the point will be clipped or not can be decided on the basis of following inequalities.

Yw_{max}

$Xw_{min} \leq X \leq Xw_{max}$

$Yw_{min} \leq Y \leq Yw_{max}$

Yw_{min}

Xw_{min} Xw_{max}

Figure 2: Point Clipping

It is to be noted that (Xw_{max}, Yw_{max}) and (Xw_{min}, Yw_{min}) can be either world coordinate window boundary or viewport boundary. Further, if any one of these four inequalities is not satisfied, then the point is clipped (not saved for display).

3.4 LINE CLIPPING

Line is a series of infinite number of points, where no two points have space in between them. So, the above said inequality also holds for every point on the line to be clipped. A variety of line clipping algorithms are available in the world of computer graphics, but we restrict our discussion to the following Line clipping algorithms, name after their respective developers:

1) Cohen Sutherland algorithm,
2) Cyrus-Beck of algorithm

3.4.1 Cohen Sutherland Line Clippings

This algorithm is quite interesting. The clipping problem is simplified by dividing the area surrounding the window region into four segments Up, Down, Left, Right (U,D,L,R) and assignment of number 1 and 0 to respective segments helps in positioning the region surrounding the window. How this positioning of regions is performed can be well understood by considering *Figure 3*.

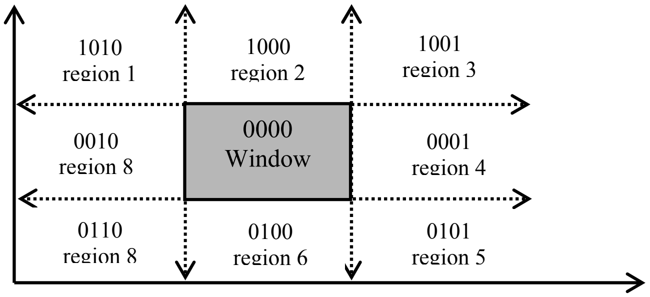


Figure 3: Positioning of regions surrounding the window

In *Figure 3* you might have noticed, that all coding of regions U,D,L,R is done with respect to window region. As window is neither Up nor Down, neither Left nor Right, so, the respective bits UDLR are 0000; now see region 1 of *Figure 3*. The positioning code UDLR is 1010, i.e., the region 1 lying on the position which is upper left side of the window. Thus, region 1 has UDLR code 1010 (Up so U=1, not Down so D=0, Left so L=1, not Right so R=0).

The meaning of the UDLR code to specify the location of region with respect to window is:

1st bit $\Rightarrow$ Up(U) ; 2nd bit $\Rightarrow$ Down(D) ;3rd bit $\Rightarrow$ Left(L) ; 4th bit $\Rightarrow$ Right(R),

Now, to perform Line clipping for various line segment which may reside inside the window region fully or partially, or may not even lie in the widow region; we use the tool of logical ANDing between the UDLR codes of the points lying on the line.

Logical ANDing (^) operation between respective bits implies => 1 ^ 1 = 1; 1 ^ 0 = 0; 0 ^ 1 = 0; 0 ^ 0 = 0

Note:

- UDLR code of window is 0000 always and w.r.t. this will create bit codes of other regions.
- A line segment is visible if both the UDLR codes of the end points of the line segment equal to 0000 i.e. UDLR code of window region. If the resulting code is not 0000 then, that line segment or section of line segment may or may not be visible.

Now, let us study how this clipping algorithm works. For the sake of simplicity we will tackle all the cases with the help of example lines l_1 to l_5 shown in *Figure 4*. Each line segment represents a case.

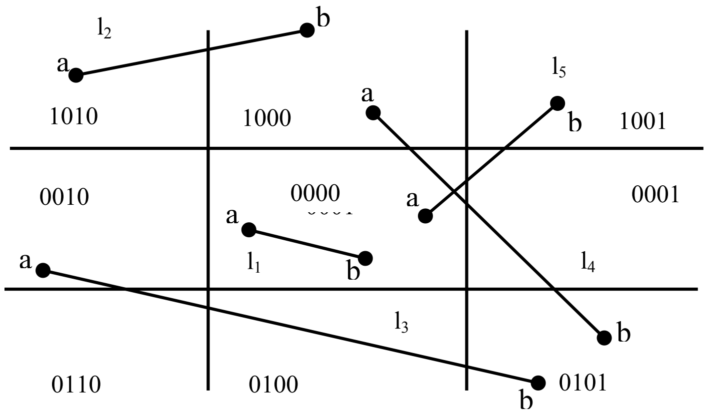


Figure 4: Various cases of Cohen Sutherland Line Clippings

Note, that in *Figure 4*, line l_1 is completely visible, l_2 and l_3 are completely invisible; $l_4$and l_5 are partially visible. We will discuss these out comings as three separate cases.

Case 1: $l_1 \rightarrow$ Completely visible, i.e., Trival acceptance
($\therefore$ both points lie inside the window)

Case 2: l_2 and $l_3 \rightarrow$ Invisible , i.e., Trival acceptance rejection

Case 3: l_4 and $l_5 \rightarrow$ partially visible ($\therefore$ partially inside the window)
Now, let us examine these three cases with the help of this algorithm:

Case 1: (Trivial acceptance case) *if the UDLR bit codes of the end points P,Q of a given line is 0000 then line is completely visible.* Here this is the case as the end points a and b of line l_1 are: a (0000), b (0000). If this trival acceptance test is failed then, the line segment PQ is passed onto Case 2.

Case 2: (Trivial Rejection Case) *if the logical intersection (AND) of the bit codes of the end points P, Q of the line segment is ≠ 0000 then line segment is not visible or is rejected.*

Note that, in *Figure 4*, line 2 is completely on the top of the window but line 3 is neither on top nor at the in bottom plus, either on the LHS nor on the RHS of the

window. We use the standard formula of logical ANDing to test the non visibility of the line segment.

So, to test the visibility of line 2 and 3 we need to calculate the logical intersection of end points for line 2 and line 3.

line l2: bit code of end points are 1010 and 1000
logical intersection of end points = (1010) ^ (1000) = 1000
as logical intersection $\neq$ 0000. So line 2 will be invisible.

line l3: end points have bit codes 0010 and 0101 now logical intersection = 0000, i.e., 0010 ^ 0101 = 0000 from the *Figure 4*, the line is invisible. Similarly in line 4 one end point is on top and the other on the bottom so, logical intersection is 0000 but then it is partially visible, same is true with line 5. These are special cases and we will discuss them in case 3.

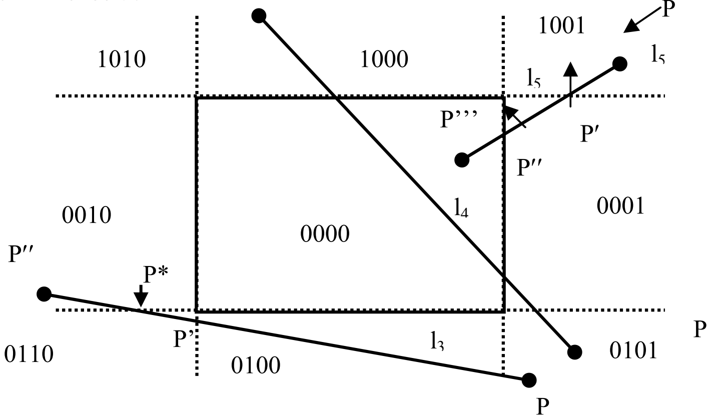


Figure 5: Cohen Sutherland line clipping

Case 3: Suppose for the line segment PQ, both the trivial acceptance and rejection tests failed (i.e., Case 1 and Case 2 conditions do not hold, this is the case for l_3, l_4 and l_5 line segments) shown above in *Figure 5*. For such non-trivial Cases the algorithm is processed, as follows.

Since, both the bitcodes for the end points P, Q of the line segment cannot be equal to 0000. Let us assume that the starting point of the line segment is P whose bit code is not equal to 0000. For example, for the line segment l_5 we choose P to be the bitcodes 1001. Now, scan the bitcode of P from the first bit to the fourth bit and find the position of the bit at which the bit value 1 appears at the first time. For the line segment l_5 it appears at the very first position. If the bit value 1 occurs at the first position then proceed to intersect the line segment with the UP edge of the window and assign the first bit value to its point of intersection as 0. Similarly, if the bit value 1 occurs at the second position while scanning the bit values at the first time then intersect the line segment PQ with the Down edge of the window and so on. This point of intersection may be labeled as P'. Clearly the line segment PP' is outside the window and therefore rejected and the new line segment considered for dipping will be P'Q. The coordinates of P' and its remaining new bit values are computed. Now, by taking P as P', again we have the new line segment PQ which will again be referred to Case 1 for clipping.

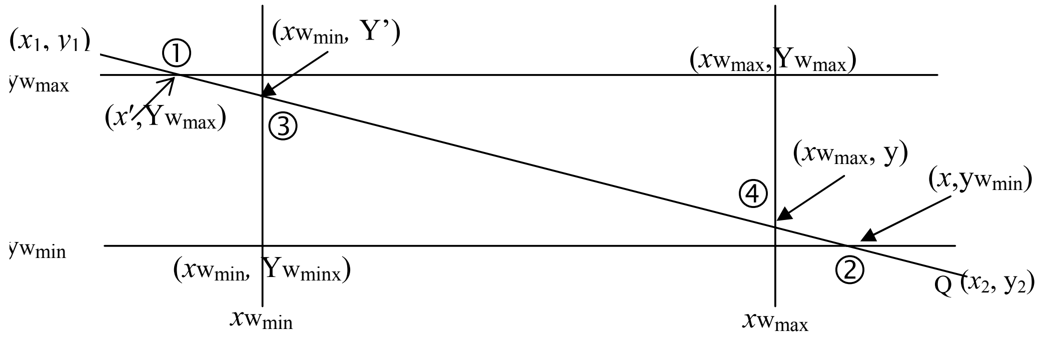


Figure 6: Line Clipping - Geometrically

Geometrical study of the above type of clipping (it helps to find point of intersection of line PQ with any edge).

Let (x_1, y_1) and (x_2, y_2) be the coordinates of P and Q respectively.

1) top case/above
   if $y_1 > yw_{max}$ then 1st bit of bit code = 1 (signifying above) else bit code = 0

2) Bottom case/below case
   if $y_1 < yw_{min}$ then 2nd bit = 1 (*i.e.* below) else bit = 0

3) Left case: if $x_1 < xw_{min}$ then 3rd bit = 1 (*i.e.* left) else 0

4) Right case: if $x_1 > xw_{max}$ then 4th bit = 1 (*i.e.* right) else 0

Similarly, the bit codes of the point Q will also be assigned.

1) **Top/above case:**

   equation of top edge is: $y = yw_{max}$. The equation of line PQ is
   $$y - y_1 = m(x - x_1)$$
   where,
   $$m = (y_2 - y_1)/(x_2 - x_1)$$
   The coordinates of the point of intersection will be (x, yw_{max}) $\therefore$ equation of line between point P and intersection point is

   $(yw_{max} - y_1) = m(x - x_1)$
   rearrange we get
   $$x = x_1 + \frac{1}{m}(yw_{max} - y_1) \qquad \text{-------------------- (A)}$$

   Hence, we get coordinates (x, yw_{max}) i.e., coordinates of the intersection.

2) **Bottom/below edge** start with $y = yw_{min}$ and proceed as for above case.

   $\therefore$ equation of line between intersection point (x', yw_{min}) and point Q *i.e.* (x_2, y_2) is
   $$(yw_{min} - y_2) = m(x' - x_2)$$

   rearranging that we get,
   $$\boxed{x' = x_2 + \frac{1}{m}(yw_{min} - y_2)} \qquad \text{-------------------- (B)}$$

   The coordinates of the point of intersection of PQ with the bottom edge will be
   $$(x_2 + \frac{1}{m}(yw_{min} - y_2), yw_{min})$$

3) **Left edge:** the equation of left edge is $x = xw_{min}$.

   Now, the point of intersection is $(xwmin, y)$.
   Using 2 point from the equation of the line we get
   $(y - y_1) = m(xw_{min} - x_1)$

   Rearranging that, we get, $y = y_1 + m(xw_{min} - x_1)$. -------------------- (C)

Hence, we get value of xw_{min} and y both *i.e.* coordinates of intersection point is given by $(xw_{min}, y_1 + m(xw_{min} - x_1))$.

4) **Right edge:** proceed as in left edge case but start with x-xw_{max}.

Now point of intersection is (xw_{max}, y′).

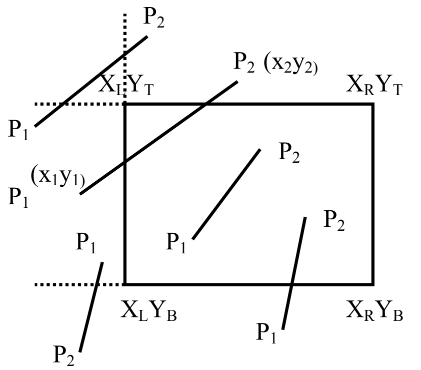
Using 2 point form, the equation of the line is

$(y' - y_2) = m\ (xw_{max} - x_2)$

$$y' = y_2 + m\ (xw_{max} - x_2) \qquad \text{--------------------} \text{(D)}$$

The coordinates of the intersection of PQ with the right edge will be $(xw_{max}, y_2 + m(xw_{max} - x_2))$.

Steps of the Cohen Sutherland Line Clipping Algorithm are



L→Left ;
R→ Right
T→Top ;
B→Bottom

Figure 7: Steps for Cohen Sutherland Line Clipping

STEP 1:

Input: $x_L, x_R, y_T, y_B, P_1(x_1, y_1), P_2(x_2, y_2)$
Initialise *i* = 1
While *i* <= 2
if $x_i < x_L$ then bit 1 of code –P_i = 1 else 0
if $x_i > x_R$ then bit 2 of code –P_i =1 else 0 : The endpoint codes of the line are set
if $y_i < y_B$ then bit 3 of code –P_i = 1 else 0
if $y_i > y_T$ then bit 4 of code –P_i = 1 else 0
i = i +1
end while
i = 1

STEP 2:

Initialise j = 1
While j <= 2
if $x_j < x_L$ then $C_{j\,left} = 1$ else $C_{j\,left} = 0$
if $x_j > x_R$ then $C_{j\,right} = 1$ else $C_{j\,right} = 0$:Set flags according to the position of the line endpoints w.r.t. window
if $y_j < y_B$ then $C_{j\,bottom} = 1$ else $C_{j\,bottom} = 0$ edges

if $y_j > y_T$ then $C_{j\,top} = 1$ else $C_{j\,top} = 0$
end while

STEP 3: If codes of $P_1$and P_2 are both equal to zero then draw P_1P_2 (totally visible)

STEP 4: If logical intersection or AND operation of code –P_1 and code –P_2 is not equal to zero then ignore P_1P_2 (totally invisible)

STEP 5: If code –P_1= 0 then swap P_1 and P_2 along with their flags and set i = 1

STEP 6: If code –P_1 < > 0 then

```
for i = 1,
{if C1 left = 1 then
find intersection (xL, y'L) with left edge vide eqn. (C)
assign code to (xL, y'L)
P1 = (xL, y'L)
end if
i = i + 1;
go to 3
}

for i = 2,
{if C1 right = 1 then
find intersection (xR, y'R) with right edge vide eqn. (D)
assign code to (xR, y'R)
P1 = (xR, y'R)
end if
i = i + 1
go to 3
}

for i = 3
{if C1 bottom = 1 then
find intersection (x'B, yB) with bottom edge vide eqn. (B)
assign code to (x'B, yB)
P1 = (x'B, yB)
end if
i = i + 1
go to 3
}

for i = 4,
{if C1 top = 1 then
find intersection (x'T, yT) vide eqn. (A) with top edge
assign code to (x'T, yT)
P1 = (x'T, yT)
end if
i = i + 1
go to 3
}
end
```

Example: A Clipping window ABCD is located as follows:

A(100, 10), B(160, 10, C(160, 40), D(100, 40). Using Sutherland-Cohen clipping algorithm find the visible portion of the line segments EF, GH and P_1P_2. E(50,0), F(70,80), G(120, 20), H(140, 80), P_1(120, 5), P_2(180, 30).

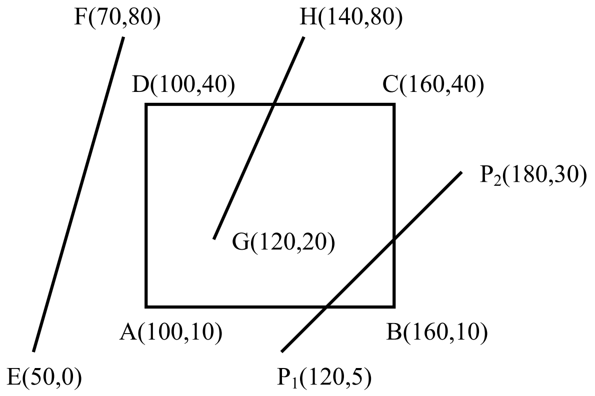


Figure 8: Example of Cohen Sutherland Line Clipping

At first considering the line P_1P_2

INPUT: P_1(120, 5), P_2(180, 30)
$x_L = 100$, $x_R = 160$, $y_B = 10$, $y_T = 40$

$x_1 > x_L$ then bit 1 of code –P_1 = 0 $C_{1\ left} = 0$
$x_1 < x_R$ then bit 2 of code –P_1 = 0 $C_{1\ right} = 0$
$y_1 < y_B$ then bit 3 of code –P_1 = 1 $C_{1\ bottom} = 1$
$y_1 < y_T$ then bit 4 of code –P_1 = 0 $C_{1\ top} = 0$

code –P_1 = 0100,

$x_2 > x_L$ then bit 1 of code –P_1 = 0 $C_{2\ left} = 0$
$x_2 > x_R$ then bit 2 of code –P_1 = 1 $C_{2\ right} = 1$
$y_2 > y_B$ then bit 3 of code –P_1 = 0 $C_{2\ bottom} = 0$
$y_2 < y_T$ then bit 4 of code –P_1 = 0 $C_{2\ top} = 0$

code –P_2 = 0010.

Both code –P_1 <> 0 and code –P_2 <> 0
then P_1P_2 not totally visible

code –P_1 AND code –P_2 = 0000
hence (code –P_1 AND code –P_2 = 0)
then line is not totally invisible.

As code –P <> 0

for i = 1
{
$C_{1\ left}$ (= 0) <> 1 then nothing is done.
i = i + 1 = 2
}
code –P_1 <> 0 and code –P_2 <> 0
then P_1P_2 not totally visible.

code –P_1 AND code –P_2 = 0000
hence (code –P_1 AND code –P_2 = 0)

then line is not totally invisible.

for i = 2

{

$C_{1\ right}$ (= 0) <> 1 then nothing is to be done.

i = i + 1 = 2 + 1 = 3

}

code –P_1 <> 0 and code –P_2 <> 0

then P_1P_2 not totally visible.

code –P_1 AND code –P_2 = 0000

Hence, (code –P_1 AND code –P_2 = 0)

then the line is not totally invisible.

for i = 3

{

$C_{1\ bottom}$ = 1 then find intersection of P_1P_2 with bottom edge

$y_B = 10$

$x_B = (180\text{-}120)(10\text{-}5)/(30\text{-}5) + 120$

$=132$

then $P_1 = (132,10)$

$x_1 > x_L$ then bit 1 of code –P_1 = 0 $C_{1\ left} = 0$

$x_1 < x_R$ then bit 2 of code –P_1 = 0 $C_{1\ right} = 0$

$y_1 = y_B$ then bit 3 of code –P_1 = 0 $C_{1\ bottom} = 0$

$y_1 < y_T$ then bit 4 of code –P_1 = 0 $C_{1\ top} = 0$

code –P_1 = 0000

i = i + 1 = 3 + 1 = 4

}

code –P_1 <> 0 but code –P_2 <> 0

then P_1P_2 not totally visible.

code –P_1 AND code –P_2 = 0000

Hence, (code –P_1 AND code –P_2 = 0)

then line is not totally invisible.

As code –P_1 = 0

Swap P_1 and P_2 along with the respective flags

$P_1 = (180, 30)$

$P_2 = (132, 10)$

code –P_1 = 0010

code –P_2 = 0000

$C_{1\ left} = 0$	$C_{2\ left} = 0$
$C_{1\ right} = 1$	$C_{2\ right} = 0$
$C_{1\ bottom} = 0$	$C_{2\ bottom} = 0$
$C_{1\ top} = 0$	$C_{2\ top} = 0$

Reset i = 1

for i = 1

{

$C_{1\ left}$ (= 0) <> 1 then nothing is to be done.

i = i + 1 = 1 + 1 = 2

}

code –P_1 <> 0, and code –P_2 <> 0

then P_1P_2 is not totally visible.

code –P_1 AND code –P_2 = 0000

Hence, (code $-P_1$ AND code $-P_2 = 0$)
then line is not totally invisible.

for i = 2
{
$C_{1\ right} = 1$ then find intersection of P_1P_2 with right edge
$x_R = 160$
$y_R = (30 - 5)(160 - 120)/(180 - 120) + 5$
$= 21.667$
$= 22$
then $P_1 = (160, 22)$
$x_1 > x_L$ then bit 1 of code $-P_1 = 0$ $C_{1\ left} = 0$
$x_1 = x_R$ then bit 2 of code $-P_1 = 0$ $C_{1\ right} = 0$
$y_1 > y_B$ then bit 3 of code $-P_1 = 0$ $C_{1\ bottom} = 0$
$y_1 < y_T$ then bit 4 of code $-P_1 = 0$ $C_{1\ top} = 0$
code $-P_1 = 0000$, i = i + 1 = 2 + 1 = 3
}

As both code $-P_1 = 0$ and code $-P_2 = 0$ then the line segment P_1P_2 is totally visible.

So, the visible portion of input line P_1P_2 is $P'_1P'_2$ where, $P_1 = (160, 22)$ and $P_2 = (132, 10)$.

Considering the line EF

1) The endpoint codes are assigned
   code – E → 0101
   code – F → 1001
2) Flags are assigned for the two endpoints
   $E_{left} = 1$ (as x coordinate of E is less than x_L)
   $E_{right} = 0$, $E_{top} = 0$ $E_{bottom} = 1$

   Similarly,
   $F_{left} = 1$, $F_{right} = 0$, $F_{top} = 1$ $F_{bottom} = 0$

3) Since codes of E and F are both not equal to zero the line is not totally visible.

4) Logical intersection of codes of E and F is not equal to zero. So, we may ignore EF line and declare it as totally invisible.

Considering the line GH

1) The endpoint codes are assigned
   code – G → 0000
   code – H → 1000

2) Flags are assigned for the two endpoints
   $G_{left} = 0$, $G_{right} = 0$, $G_{top} = 0$, $G_{bottom} = 0$.

   Similarly,
   $H_{left} = 0$, $H_{right} = 0$, $H_{top} = 1$, $H_{bottom} = 0$.

3) Since, codes of G and H are both not equal to zero so the line is not totally visible.

4) Logical intersection of codes of G and H is equal to zero so we cannot declare it as totally invisible.

5) Since, code – G = 0, Swap G and H along with their flags and set i = 1

Implying $G_{left} = 0$, $G_{right} = 0$, $G_{top} = 1$, $G_{bottom} = 0$.

$H_{left} = 0$, $H_{right} = 0$, $H_{top} = 0$, $H_{bottom} = 0$.
as G → 1000, H → 0000

6) Since, code – G <> 0 then

for i = 1, {since $G_{left} = 0$
i = i + 1 = 2
go to 3
}

The conditions 3 and 4 do not hold and so we cannot declare line GH as totally visible or invisible.

for i = 2, {since $G_{right} = 0$
i = i + 1 = 3
go to 3
}

The conditions 3 and 4 do not hold and so we cannot declare line GH as totally visible or invisible.

for i = 3, {since $G_{bottom} = 0$
i = i + 1 = 4
go to 3
}

The conditions 3 and 4 do not hold and so we cannot declare line GH as totally visible or invisible.

for i = 4, {since $G_{top} = 1$
Intersection with top edge, say P(x, y) is found as follows:

Any line passing through the points G, H and a point P(x, y) is given by

$y – 20 = \{(180 – 20) /(140 – 120)\}(x – 120)$
or, $y – 20 = 3x – 360$
or, $y – 30 = -340$

Since, the y coordinate of every point on line CD is 40, so we put y = 40 for the point of intersection P(x, y) of line GH with edge CD.

$40 – 3x = -340$
or, $- 3x = - 380$
or $x = 380/3 = 126.66 \approx 127$

So, the point of intersection is P(127, 40). We assign code to it.

Since, the point lies on edge of the rectangle so the code assigned to it is 0000.

Now, we assign G = (127, 40); i = 4 + 1 = 5. Conditions 3 and 4 are again checked.

Since, codes G and H are both equal to 0, so, the line between H(120, 20) and G(127, 40) is totally visible.

Limitation of Cohen Sutherland line Clipping Algorithm

The algorithm is only applicable to rectangular windows and not to any other convex shaped window.

So, a new line-clipping algorithm was developed by Cyrus, and Beck to overcome this limitation. This Cyrus Beck line-clipping Algorithm is capable of clip lining segments irrespective of the shape of the convex window.

You might wonder as to what a convex window? In general, on the basis of the shapes the windows are classified into two categories:

i) **Convex shaped windows:** Windows of a shape such that if we take any two points inside the window then the line joining them will never cross the window boundary, such shaped windows are referred as convex shaped windows.

ii) **Concave or non Convex shaped windows:** Windows of a shape such that if we can choose a pair of points inside the window such that the line joining them may cross the window boundary or some section of the line segment may lie outside the window. Such shaped windows are referred to as non-convex or concave shaped windows.

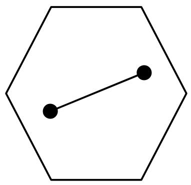


Figure (a)
Convex Polygonal Window

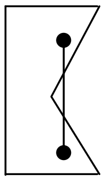


Figure (b)
Non-convex Polygonal window

Figure 9: Types of Windows

3.4.2 Cyrus-Beck Algorithm

Cyrus Beck Line clipping algorithm is infact, a parametric line-clipping algorithm. The term parametric implies that we need to find the value of the parameter t in the parametric representation of the line segment for the point at which the segment intersects the clipping edge. For better understanding, consider the *Figure 9(a),* where P Q is a line segment, which is intersecting at the two edges of the convex window.

Note: The algorithm is applicable to the "convex polygonal window".

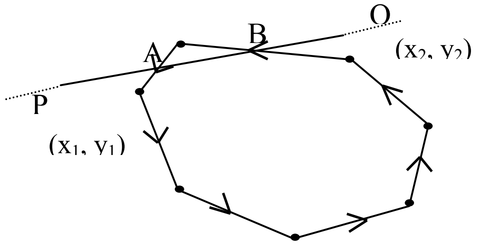


Figure 9(a): Interaction of line PQ and Window

Now, just recall the parametric equation of line segment PQ, which we have studied in the course CS-60.

It is simply $P + t(Q - P) \quad 0 \le t \le 1$

Where, $t \rightarrow$ linear parameter continuously changes value.

$\therefore P + t(Q - P) \Rightarrow (x_1, y_1) + t(x_2 - x_1, y_2 - y_1) = (x, y)$ be any point on PQ. ------ (1)

For this equation (1) we have following cases:

1) When $t = 0$ we obtain the point P.
2) When $t = 1$ we get the point Q.
3) When t varies such that $0 \le t \le 1$ then line between point P and Q is traced.

For t = ½ we get the mid-point of PQ.

4) When t < 0 line on LHS of P is traced.
5) When t > 1 line on RHS of Q is traced.

So, the variation in parameter t is actually generating line in point wise manner .The range of the parameter values will identify the portion to be clipped by any convex polygonal region having n-vertices or lattice points to be specified by the user. One such clipping situation is shown in *Figure 9(a).*

Remark:

- **How to specify the window region:** a convex polygonal region having n-vertices $\{P_0, P_1, P_2\ldots, P_{n-1}, P_n, P_0\}$ or lattice points to be specified by the user encloses the convex window region. To be specific about the window we may say that each edge between two points contributes to the boundary of the window under the situation that (when we traverse each edge in anticlockwise manner), the window region lies on left hand side (LHS) of edge. This orientation is preserved and while giving input, the user takes care of the orientation to specify the window region of any arbitrary convex shape.

- **Potentially entering and leaving points (P_E and P_L)**
The point of intersection of the line and window may be classified either as Potentially entering or leaving. Before going into other details, let us describe the actual meaning of Potentially entering and leaving points (P_E and P_L). P_E and P_L are dependent on the edge orientation, *i.e.* direction. Let us understand **how to find P_E and P_L** (we know as we move anticlockwise across the window boundary then region of LHS encloses the window).

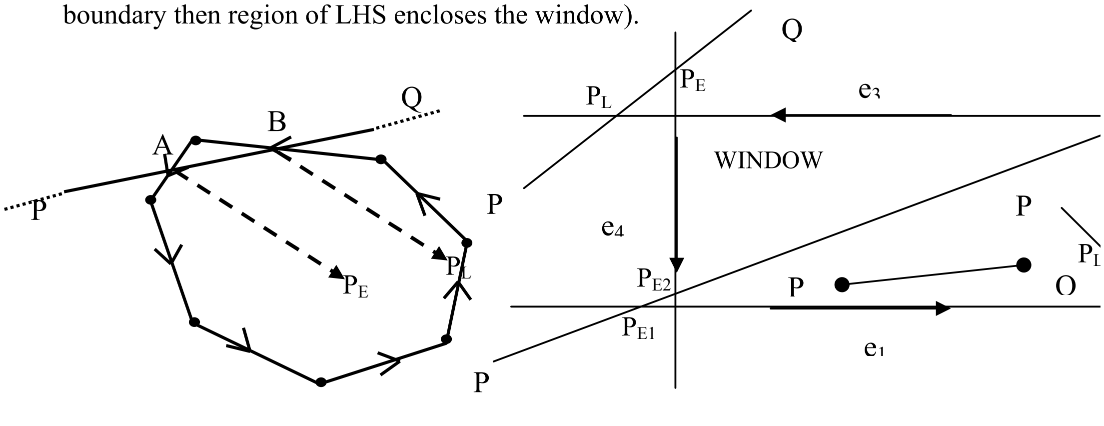


Figure 10(a): Potentially Entering P_E and Potentially Leaving P_L points

Say, we are moving from point P to point Q, and we know that while traversing the windows edges in anticlockwise manner the region lying on the left hand side is referred to as the window region. So, from the *Figure 10 (a)*, you may notice that at point P_E we are moving from P to Q we are entering the window region , thus this point of intersection should be referred to as the Potentially entering point(P_E). Now, refer to other intersection shown in the *Figure 10 (a)*, here, also the movement of points on the line are determined by varying value of parameter t is from point P to Q and w.r.t ., the orientation of window boundary, we are exiting the window region. So, this point of intersection is known as potentially leaving point (P_L).

Potentially entering point (P_E) while moving towards the window, the point of intersection between line PQ and the window edges faced are known as **P_E**.

Potentially leaving point (P_L) These are the point of intersection encountered while we move away from window.

OR

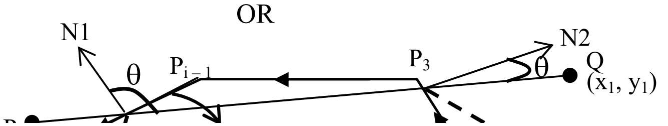


Figure 10 (b): Potentially Entering P_E and Potentially Leaving P_L points

Other way round you may detect the point of intersection as Potentially leaving point (P_L) or Potentially entering point (P_E) by studying the angle between the line to be clipped from the outward normal to the intersecting edge, at the point of intersection. From *Figure 10 (b)*, it is the angle between PQ and N1 or N2. You may notice that, while moving from P to Q the angle between line PQ and N1 is obtuse whereas the angle between PQ and N2 is acute; so, you may say, if the angle between line PQ and N1 is obtuse the point of intersection is potentially entering (P_E) whereas, if the angle between PQ and N2 is acute, then the point of intersection is potentially leaving (P_L).

OR

P_E => **$N_i \cdot PQ < 0$; (angle θ greater than 90° or Obtuse)**
$N_i \cdot (Q-P) < 0$

P_L => **$N_i \cdot PQ > 0$; (angle θ is less than 90° or Acute)**
$N_i \cdot (Q-P) > 0$

- **Where N_i is the outward normal to the ith edge.**

Note: when $(\overline{Q} - \overline{P}) \cdot \overline{N_i} = 0$ then it implies that:

1) $\overline{Q} - \overline{P} = 0$ ↓ not possible ∵ if $\overline{Q} - \overline{P} = 0$ then $\overline{PQ}$ is a point and not a line

2) $\overline{N_i} = 0$ ↓ not possible

3) $\theta = 90°$ ↓ possible i.e. $(\overline{Q} - \overline{P}) \perp \overline{N_i}$ ⇩ line segment PQ is || to i^{th} edge then only $\overline{N_i}$ will be ⊥ to both PQ and i^{th} edge.

- **How to find the normal :**

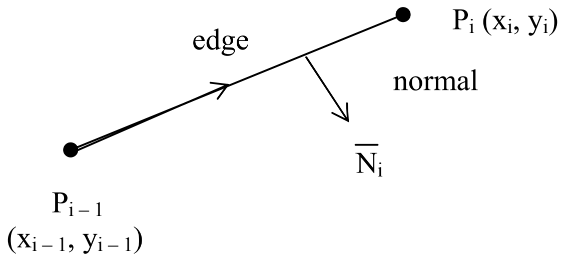


Figure 11 : Normal Determination

$$\overrightarrow{P_{i-1}P_i} = (x_1 - x_{i-1}, y_i - y_{i-1}) = (s_1, s_2)$$

if $\overrightarrow{N_i} = (n_{1i}, n_{2i})$, then

$$\overrightarrow{N_i} \cdot \overrightarrow{P_{i-1}P_i} = 0 \Rightarrow (s_1, s_2) \cdot (n_{1i}, n_{2i}) = 0$$

$$\Rightarrow s_1 n_{1i} + s_2 n_{2i} = 0$$
$$\Rightarrow s_1 n_{1i} = - s_2 n_{2i} = 0$$
$$\Rightarrow n_{1i} = \frac{-s_2}{s_1} n_{2i}$$

If $n_{2i} = 1$ then, $n_{1i} = \frac{s_2}{s_1}$

Therefore, $\overrightarrow{N_i} = (n_{1i}, n_{2i}) = (\frac{-s_2}{s_1}, 1) \longrightarrow$ NORMAL

if $\left(\frac{-s_2}{s_1}, 1\right)$ is outward normal then $-\left(\frac{-s_2}{s_1}, 1\right)$ i.e. $\left(\frac{s_2}{s_1}, -1\right)$ is inward normal.

Now, we have learned some basic things which will be required in the working of the algorithm, so, we can start using the concepts learned so far to clip the line segment passing through the window.

We know that the parametric equation of line PQ is

$$P + t\,(Q - P) \;\; ; \;\; 0 \le t \le 1$$

Now, to find the point of intersection of line PQ and i^{th} edge we need to find the appropriate value of parameter t, for that we firstly find normal to the ith edge. Say $\overrightarrow{Ni}$ is the normal to i^{th} edge (P_{i-1} to P_i), then to find the appropriate parameter t, we should determine the dot product of the vector from P_{i-1} to the point of intersection and the normal Ni. Let us do this exercise.

The vector joining edge point P_{i-1} and point of intersection of the line PQ with the ith edge for some t will be given by:

$$\{ [\overline{P} + t(\overline{Q} - \overline{P})] - \overline{P}_{i-1} \}$$

As $\overrightarrow{Ni}$ is normal to the ith edge , so, the dot product of the vector joining edge point P_{i-1} and point of intersection of the line PQ with the ith edge for some t and Ni will be given by:

$$\{ [\overline{P} + t(\overline{Q} - \overline{P})] - \overline{P}_{i-1} \} . \overrightarrow{N_i} = 0 \qquad \text{--------------(1)}$$

Rearranging (1) we get,

$$t = \frac{-(\overline{P} - \overline{P}_{i-1}) . \overline{N_i}}{(\overline{Q} - \overline{P}) . \overline{N_i}} \qquad \text{---------------(2)}$$

Using this value of t from (2) in parametric form of line we will get the point of intersection of line PQ and i^{th} edge.

Note: By the above discussion of P_E and P_L we came to know that if $N_i . (Q-P) < 0 \Rightarrow P_E$ point and $N_i . (Q-P) > 0 \Rightarrow P_L$ point. If the value of t calculated using (2) lies in the interval 0 to 1 (in magnitude) then, the point of intersection will be considered otherwise it will not be considered.

Steps to clip a line segment PQ:

- Firstly, find all the points of intersections of the line segment PQ with the edges of the polygonal window and classify them either as P_E and P_L points. Also

determine the value of parmeter t , using equation (2) for respective PE's and PL's.

Or

If value of the normals to respective edges are not given then, find the value of normal to every edge of the window, then, determine the value of parmeter t , using equation (2) for respective point of intersection between line segment and window edge then on the basis of the vaue of parameter t mark them as $\mathbf{P_E}$ and $\mathbf{P_L}$ provided the value of t lies from 0 to 1 in magnitude.

- Secondly, out of the various values of t for $\mathbf{P_E}$'s determine the maximum value of t say it be t_{max}. Similarly, out of the various values of t for $\mathbf{P_L}$'s determine the minimum value of t say it be t_{min} . Note that for clipping to be possible $t_{min} > t_{max}$.

- Finally, vary the parameter t from t_{max} to t_{min} and determine the clipped line as outcome.

For better understanding of steps consider *Figure 12*.

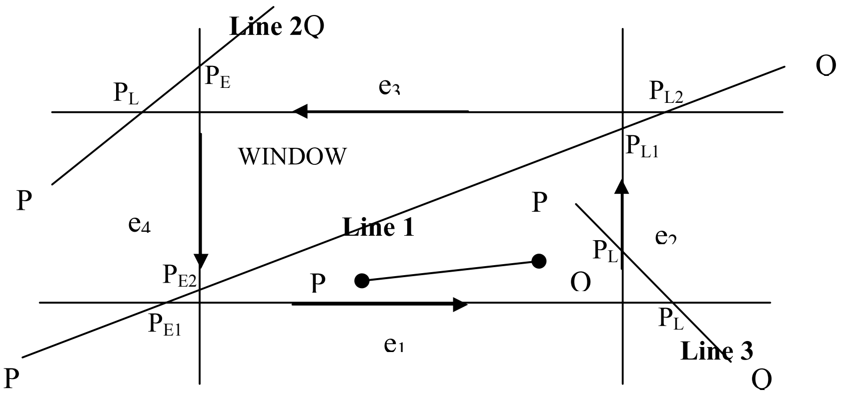


Figure 12: Steps of Cyrus Beck Clipping

In case of Line 1: (PQ)

1) Point 1 is potentially entering (P_{E_1}) because as we move along PQ, the LHS of e_1 is window and hence, it seems that we are entering the window.
2) Point 2 is again P_{E_2} similarly as point 1.
3) Point 3 is potentially leaving (P_4) ∵ as we move along PQ, the LHS of e_2 is window and hence, it seems that we are leaving the window.
4) Similarly, point 4 is also PL_2.

Line 2 and 3 (PQ):

Using same logic as for line 1 we find $\overline{P}_L$ and P_E. Now, it is to be noted that for each point of intersection we have some value of t.

say t_1 is value of t for P_{E_1}
say t_2 is value of t for P_{E_2}
say t_3 is value of t for P_{L_1}
say t_4 is value of t for P_{L_2}

As the portion visible to us is image what laying inside the window for line 1 the line visible is from P_{E_2} to P_{L_1} and for these points there is some value of t. With this initialisation let us study the following cases:

Case 1: Say we get a new value of t_E (i.e value of parameter t for any potentially entering (P_E) point) we choose t_{max} as: $t_{max} = \max \{t_{max}, t_E\}$. The initial value of $t_{max} = 0$ is taken.

Case 2: Say we get a new value of t_L (i.e value of parameter t for any potentially leaving(P_L) point) then t_{max} value is updated by $t_{min} = \min \{t_{min}, t_L\}$. The initial value of $t_{min} = 1$ is taken.

Finally when value of t for all edges are found and if $t_{max} < t_{min}$ then line is visible else not. And line is visible from

$P + t_{max} (Q - P)$ to $P + t_{min} (Q - P)$ *i.e.*

$P + t (Q - P)$ such that $t_{max} \le t \le t_{min}$

$t_{max} < t_{min}$ (draw line)

$t_{max} \nless t_{min}$ (reject line)

Example: How does the Cyrus Beck line clipping algorithm, clip a line segment if the window is non convex?

Solution: Consider the *Figure 13*, here, the window is non convex in shape and PQ is a line segment passing through this window .Here too the condition of visibility of the line is $t_{max} < t_{min}$ and the line is visible from $P + t_{max} (Q - P)$ to $P + t_{min} (Q - P)$, if $t_{max} \nless t_{min}$ then reject the line segment. Now, applying this rule to the *Figure 13,* we find that when PQ line segment passes through the non convex window, it cuts the edges of the window at 4 points. $1 \rightarrow P_E$; $2 \rightarrow P_L$; $3 \rightarrow P_E$; $4 \rightarrow P_L$. In this example, using the algorithm we reject the line segment PQ but it is not the correct result.

Condition of visibility is satisfied in region 1-2 and 3-4 only so the line exists there but in region 2-3 the condition is violated so the line does not exists.

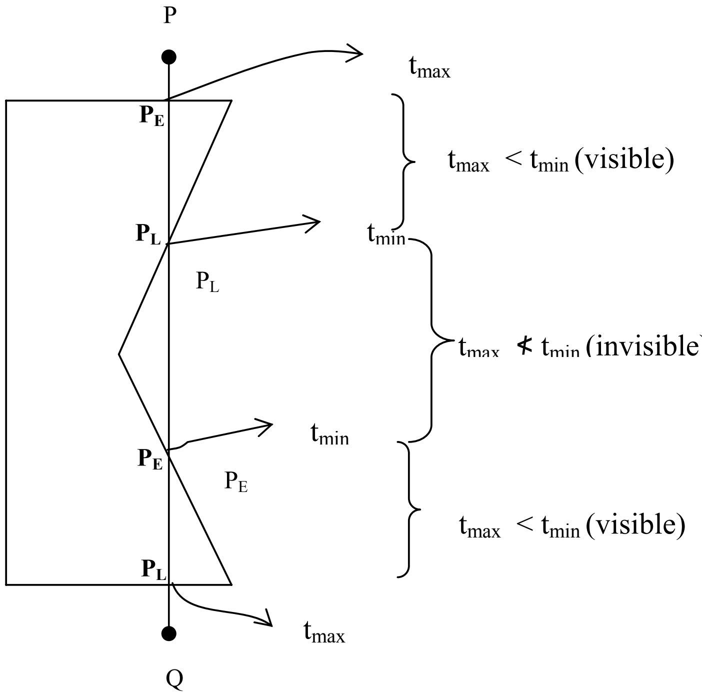


Figure 13: Example Cyrus Beck Clipping

☞ Check Your Progress 1

1) Suppose a rectangular window ABCD is defined such that A = (–1, –2) and C(3, 1) using generalised geometrical approach , clip the line segment joining the points P(–20, 0) and Q(20, 30).

………………………………………………………………………………………

………………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

…………

2) A clipping window is given by $P_0(10,10)$, $P_1(20,10)$, $P_2(25,15)$, $P_3(20,20)$, $P_4(10,15)$, $P_5 = P_0$. Using Cyrus Beck algorithm clip the line A(0,0) and B(30,25).

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

3) How will you compute the outward normal to a clipping in the case of Cyrus-Back algorithm?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

4) Compare Cohen Sutherland Line clipping with Cyrus Beck Line clipping algorithm?

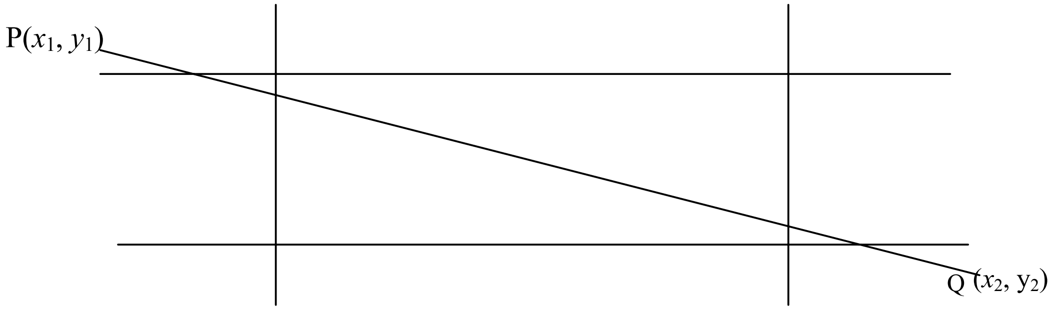
……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

5) Clip the line shown in Figure below using Cohen Sutherland Line clipping algorithm.



……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

6) Which type of clipping windows can't be handled by Cyrus Beck clipping algorithm, how can such cases be handled?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

3.5 POLYGON CLIPPING

After understanding the concept of line clipping and its algorithms, we can now extend the concept of line clipping to polygon clipping, because polygon is a surface enclosed by several lines. Thus, by considering the polygon as a set of line we can divide the problem to line clipping and hence, the problem of polygon clipping is simplified. But it is to be noted that, clipping each edge separately by using a line clipping algorithm will certainly not produce a truncated polygon as one would expect. Rather, it would produce a set of unconnected line segments as the polygon is exploded. Herein lies the need to use a different clipping algorithm to output truncated but yet bounded regions from a polygon input. Sutherland-Hodgman algorithm is one of the standard methods used for clipping arbitrary shaped polygons with a rectangular clipping window. It uses divide and conquer technique for clipping the polygon.

3.5.1 Sutherland-Hodgman Algorithm

Any polygon of any arbitrary shape can be described with the help of some set of vertices associated with it. When we try to clip the polygon under consideration with any rectangular window, then, we observe that the coordinates of the polygon vertices satisfies one of the four cases listed in the table shown below, and further it is to be noted that this procedure of clipping can be simplified by clipping the polygon edgewise and not the polygon as a whole. This decomposes the bigger problem into a set of subproblems, which can be handled separately as per the cases listed in the table below. Actually this table describes the cases of the Sutherland-Hodgman Polygon Clipping algorithm.

Thus, in order to clip polygon edges against a window edge we move from vertex V_i to the next vertexV_{i+1} and decide the output vertex according to four simple tests or rules or cases listed below:

Table: Cases of the Sutherland-Hodgman Polygon Clipping Algorithm

Case	V_i	V_{i+1}	Output Vertex
A	Inside window)	Inside	V_{i+1}
B	Inside	Outside	V'_i i.e intersection of polygon and window edge
C	Outside	Outside	None
D	Outside	Inside	V'_I ; V_{i+1}

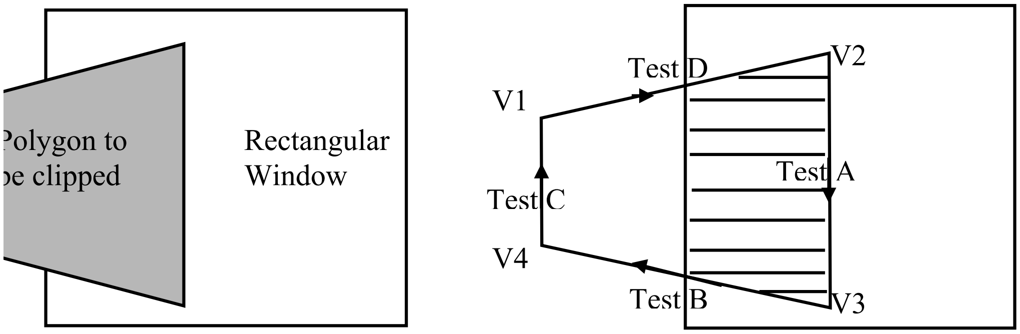
In words, the 4 possible Tests listed above to clip any polygon states are as mentioned below:

1) If both Input vertices are inside the window boundary then only 2nd vertex is added to output vertex list.
2) If 1st vertex is inside the window boundary and the 2nd vertex is outside then, only the intersection edge with boundary is added to output vertex.
3) If both Input vertices are outside the window boundary then nothing is added to the output list.
4) If the 1st vertex is outside the window and the 2nd vertex is inside window, then both the intersection points of the polygon edge with window boundary and 2nd vertex are added to output vertex list.

So, we can use the rules cited above to clip a polygon correctly. The polygon must be tested against each edge of the clip rectangle; new edges must be added and existing edges must be discarded , retained or divided. Actually this algorithm decomposes

the problem of polygon clipping against a clip window into identical subproblems, where a subproblem is to clip all polygon edges (pair of vertices) in succession against a single infinite clip edge. The output is a set of clipped edges or pair of vertices that fall in the visible side with respect to clip edge. These set of clipped edges or output vertices are considered as input for the next sub problem of clipping against the second window edge. Thus, considering the output of the previous subproblem as the input, each of the subproblems are solved sequentially, finally yielding the vertices that fall on or within the window boundary. These vertices connected in order forms, the shape of the clipped polygon.

For better understanding of the application of the rules given above consider the *Figure 14*, where the shaded region shows the clipped polygon.



(a) (b)

Figure 14 :Sutherland-Hodgman Polygon Clipping

Pseudocode for Sutherland – Hodgman Algorithm

Define variables

inVertexArray is the array of input polygon vertices
outVerteArray is the array of output polygon vertices
Nin is the number of entries in inVertexArray
Nout is the number of entries in outVertexArray
n is the number of edges of the clip polygon
ClipEdge[x] is the xth edge of clip polygon defined by a pair of vertices
s, p are the start and end point respectively of current polygon edge
i is the intersection point with a clip boundary
j is the vertex loop counter

Define Functions

AddNewVertex(newVertex, Nout, outVertexArray)
: Adds newVertex to outVertexArray and then updates Nout

InsideTest(testVertex, clipEdge[x])
: Checks whether the vertex lies inside the clip edge or not; retures TRUE is inside else returns FALSE

Intersect(first, second, clipEdge[x])
: Clip polygon edge(first, second) against clipEdge[x], outputs the intersection point

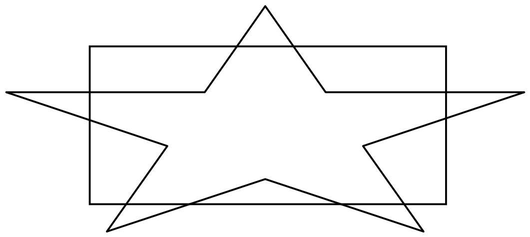
```
{                          : begin main
x = 1
while (x ≤ n)              : Loop through all the n clip edges
{
Nout = 0                   : Flush the outVertexArray
s = inVertexArray[Nin] : Start with the last vertex in inVertexArray
for j = 1 to Nin do        : Loop through Nin number of polygon vertices (edges)
        {
                p = inVertexArrray[j]
```

```
        if InsideTest(p, clipEdge[x] = = TRUE then                : Case A
and D
        if InsideTest(s, clipEdge[x] = = TRUE then
        AddNewVertex(p, Nout, outVertexArray)                     :  Case A
          else
        i = Intersect(s, p, clipEdge[x])                       :  Case D
        AddNewVertex(i, Nout, outVertexArray)
        AddNewVertex(p, Nout, outVertexArray)
        end if
        else              :  i.e. if InsideTest(p, clipEdge[x] = = FALSE
                          (Cases 2 and 3)
        if InsideTest(s, clipEdge[x]) = =TRUE then                : Case B
        {
          Intersect(s, p, clipEdge[x])
          AddNewVertex(i, Nout, outVertexArray)
          end if                          : No action for case C

          s = p                           : Advance to next pair of vertices
          j = j + 1
         end if                           :  end {for}
           }
        x = x + 1                         : Proceed to the next ClipEdge[x +1]
        Nin = Nout
        inVertexArray = outVertexArray            :  The ouput vertex array for
                                    the current clip edge becomes the input
                                    vertex array for the next clip edge
        }                           : end while
        }                           : end main
```

☞ **Check Your Progress 2**

1) Using Sutherland-Hodgeman polygon clipping algorithm clip on the polygon given below.



3.6 WINDOWING TRANSFORMATIONS

From section 3.1,we understood the meaning of the term window and viewport which could again be understood as:

Window: A world coordinate area selected for display (i.e. area of picture selected for viewing).

Viewport: An Area or a display device to which a window is mapped.

Note:

- Window defines what is to be viewed and viewpoint defines where it is to be displayed.
- Often window and viewpoints are rectangles in standard position with edges parallel to coordinate axes. Generalised shapes like polygon etc., take long to

process, so we are not considering these cases where window or viewport can have general polygon shape or circular shape.

The mapping of a part of a world coordinate scene to device coordinates is referred to as Viewing Transformation. In general 2D viewing transformations are referred to as window to viewport transformation or windowing transformation.

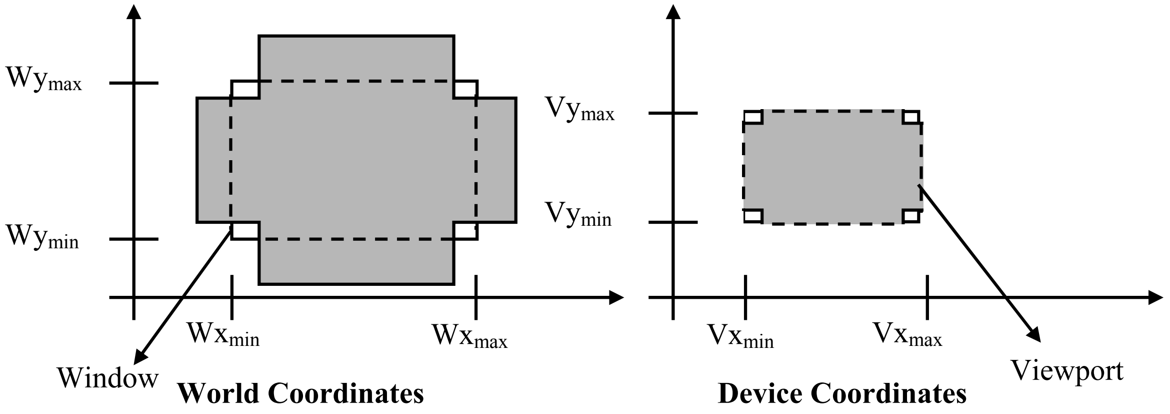


Figure 15: Windowing Transformation

You may notice in *Figure 15*, that all parts of the picture that lie outside the window are clipped and the contents which lie inside the widow are transferred to device coordinates. Secondly, you may also notice that while window selects a part of the scene, viewport displays the selected part at the desired location on the display area. When window is changed we see a different part of the scene at the same portion (viewport) on display. If we change the viewport only, we see the same part of the scene drawn at a different scale or at a different place on the display. By successively increasing or decreasing the size of the window around a part of the scene the viewport remain fixed, we can get the effect of zoom out or zoom in respectively on the displayed part. Mathematically, viewing transformation can be expressed as V=W.N

Where,

- **V** refers Viewing transformation which maps a part of world coordinate scene to device coordinates;
- **W** refers to workstation transformation which maps normalised device coordinates to physical device coordinates;
- **N** refers to Normalisation transformation used to map world coordinates to normalized device coordinates.

Window to Viewpoint Coordinates transformation:

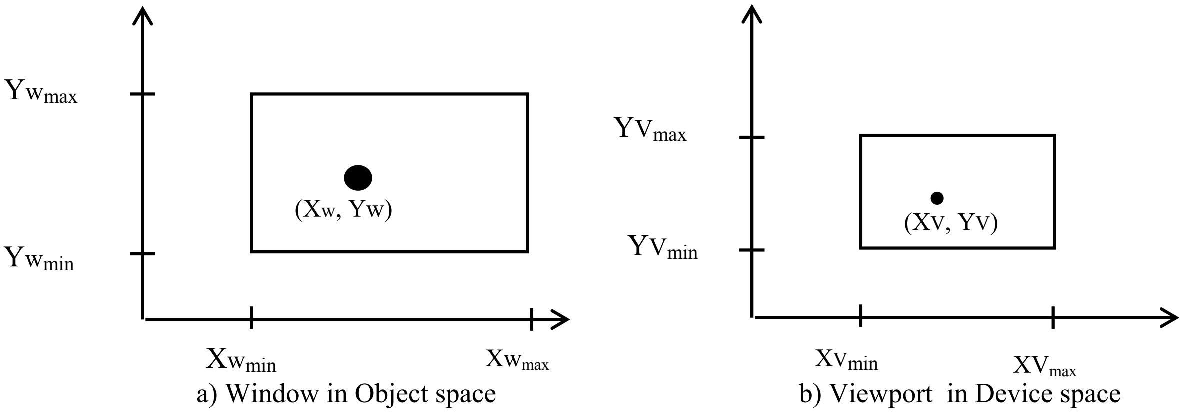


Figure 16: Window to Viewport Transformation

Figure 16, illustrates window-viewpoint mapping. Here, it is depicted that a point at position (X_w, Y_w) in window is mapped into position (X_v, Y_v) in the associated viewpoint.

So, as to maintain the same relative placement in the viewpoint as in the window we require

$$\left.\begin{array}{l}\dfrac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \dfrac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} \quad \text{-------------(a)} \\[2ex] \dfrac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \dfrac{yw - yw_{\min}}{yw_{\max} - yw_{\min}} \quad \text{-------------(b)}\end{array}\right\} \quad \text{-------------(1)}$$

rearranging equation (a) and (b) of (1) we get viewpoint position (x_v, y_v) i.e.,

$$\left(\begin{array}{l} xv = xv_{\min} + (xw - xw_{\min})\, S_x \\[1ex] yv = yv_{\min} + (yw - yw_{\min})\, S_y \end{array}\right) \quad \text{------------(2)}$$

where

$$\left.\begin{array}{ll} S_x \text{ scaling factor along } x \text{ axis} = & \dfrac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}} \\[2ex] S_y \text{ scaling factor along y axis} = & \dfrac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}} \end{array}\right\} \quad \text{----------(3)}$$

Note, if $S_x = S_y$ then the relative proportions of objects are maintained else the world object will be stretched or contracted in either *x* or *y* direction when displayed on output device.

In terms of Geometric transformations the above relation can be interpreted through the following two steps:

Step 1 Scaling the window area to the size of the viewport with scale factors s_x and s_y w.r.t a fixed point ($xw_{\min}$, $yw_{\min}$).

$$\Rightarrow [T_1] = \begin{pmatrix} s_x & 0 & xw_{\min}(1-s_x) \\ 0 & s_y & yw_{\min}(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}$$

Step 2 Translating the scaled window to the position of the viewport so that,

$$\Delta x = xv_{\min} - xw_{\min} \quad \text{and}$$
$$\Delta y = yv_{\min} - yw_{\min}$$

$$\Rightarrow [T_2] = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}$$

Concatenating $[T_1]$ and $[T_2]$ we get,

$$[T] = [T_1]\,[T_2] = \begin{pmatrix} s_x & 0 & \Delta x + xw_{\min}(1-s_x) \\ 0 & s_y & \Delta y + yw_{\min}(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}$$

Replacing the values of Δx and Δy in the above transformation matrix we finally get,

$$[T] = \begin{pmatrix} s_x & 0 & -s_x xw_{\min} + xv_{\min} \\ 0 & s_y & -s_y yw_{\min} + yv_{\min} \\ 0 & 0 & 1 \end{pmatrix} \qquad (1)$$

The aspect ratio of a rectangular window or viewport is defined by

$$a = \frac{x_{max} - x_{min}}{y_{max} - y_{min}}$$

If $s_x = s_y$ then $\dfrac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = \dfrac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$

$$\Rightarrow \frac{xv_{max} - xv_{min}}{yv_{max} - yv_{min}} = \frac{xw_{max} - xw_{min}}{yw_{max} - yw_{min}} \Rightarrow a_v = a_w$$

So, it can be said that if the aspect ratio a_v of the viewport equals the aspect ratio a_w of the window, then $s_x = s_y$ and no distortion of displayed scene occurs other than uniform magnification or compression. If $a_v \neq a_w$ then the displayed scene in the viewport gets somewhat distorted w.r.t the scene captured by the window.

Example Find the normalisation transformation *N* which uses the rectangle W (1, 1), X (5, 3), Y (4, 5) and Z (0, 3) as a window and the normalised device screen as the viewport.

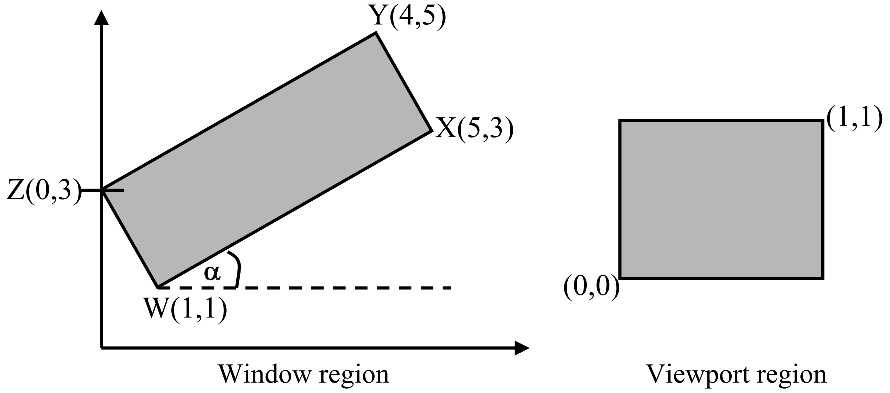


Figure 17: Example Transformations

Here, we see that the window edges are not parallel to the coordinate axes. So we will first rotate the window about W so that it is aligned with the axes.

$$\text{Now, } \tan\alpha = \frac{3-1}{5-1} = \frac{1}{2}$$

$$\Rightarrow \sin\alpha = \frac{1}{\sqrt{5}} \quad \cos\alpha = \frac{2}{\sqrt{5}}$$

Here, we are rotating the rectangle in clockwise direction. So α is (–)ve i.e., – α

The rotation matrix about W (1, 1) is,

$$[T_{R.\theta}]_W = \begin{bmatrix} \mathbf{Cos\ \alpha} & \mathbf{-Sin\ \alpha} & \mathbf{(1\text{-}Cos\ \alpha)\ x_p + Sin\ \alpha\ \ y_p} \\ \mathbf{Sin\ \alpha} & \mathbf{Cos\ \alpha} & \mathbf{(1\text{-}Cos\ \alpha)\ y_p - Sin\ \alpha\ \ x_p} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$[T_{R.\theta}]_W = \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \left(\frac{1-3}{\sqrt{5}}\right) \\ \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & \left(\frac{1-1}{\sqrt{5}}\right) \\ 0 & 0 & 1 \end{pmatrix}$$

The *x* extent of the rotated window is the length of WX which is $\sqrt{(4^2 + 2^2)} = 2\sqrt{5}$

Similarly, the y extent is length of WZ which is $\sqrt{(1^2 + 2^2)} = \sqrt{5}$

For scaling the rotated window to the normalised viewport we calculate s_x and s_y as,

$$s_x = \frac{\text{viewport } x \text{ extent}}{\text{window } x \text{ extent}} = \frac{1}{2\sqrt{5}}$$

$$s_y = \frac{\text{viewport } y \text{ extent}}{\text{window } y \text{ extent}} = \frac{1}{\sqrt{5}}$$

As in expression (1), the general form of transformation matrix representing mapping of a window to a viewport is,

$$[T] = \begin{pmatrix} s_x & 0 & -s_x xw_{\min} + xv_{\min} \\ 0 & s_y & -s_y yw_{\min} + yv_{\min} \\ 0 & 0 & 1 \end{pmatrix}$$

In this problem $[T]$ may be termed as N as this is a case of normalisation transformation with,

$$xw_{\min} = 1 \qquad xv_{\min} = 0$$
$$yw_{\min} = 1 \qquad yv_{\min} = 0$$
$$s_x = \frac{1}{2\sqrt{5}} \qquad s_y = \frac{1}{\sqrt{5}}$$

By substituting the above values in $[T]$ i.e. N,

$$N = \begin{pmatrix} \frac{1}{2\sqrt{5}} & 0 & \left(\frac{-1}{2}\right)\frac{1}{\sqrt{5}} + 0 \\ 0 & \frac{1}{\sqrt{5}} & \left(\frac{-1}{\sqrt{5}}\right)1 + 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Now, we compose the rotation and transformation N to find the required viewing transformation N_R

$$N_R = N\,[T_{R.\theta}]_W = \begin{pmatrix} \frac{1}{2\sqrt{5}} & 0 & \frac{-1}{2\sqrt{5}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{5} & 1 - \frac{3}{\sqrt{5}} \\ \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 1 - \frac{1}{\sqrt{5}} \\ 0 & 0 & 1 \end{pmatrix}$$

3.7 SUMMARY

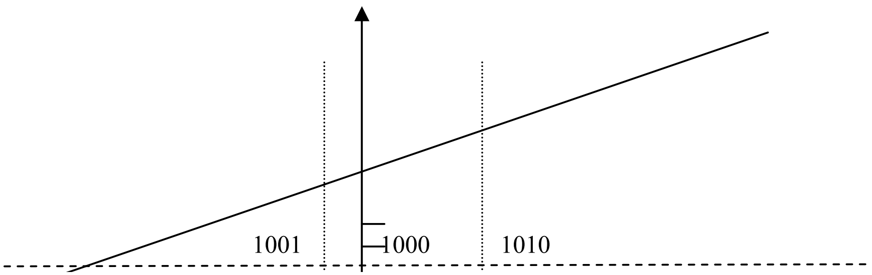
In this unit, we have discussed the concept of clipping, along with the concept of clipping we have also discussed various clipping types. The unit is quite important from the point of view of achieving realism through computer graphics.

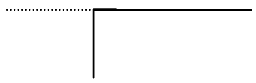
This unit contains algorithms, which are easy to implement in any programming language.

3.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Rectangular window ABCD is defined such that A = (–1, –2) and C (3, 1). So, to clip the line segment joining the points P(–20, 0) and Q(20, 30) using generalised geometrical approach we do the following tasks.





Equations of line segment PQ is:

y = mx + c where $m = \frac{y_2 - y_1}{x_2 - x_1}$ i.e $m = \frac{30 - 0}{20 - (-20)} = \frac{30}{40} = 3/4$

$\therefore$ equation of line PQ $\Rightarrow$ y = 3/4x + c ------------(1)

As point P (-20, 0) lies on the line given by equation (1) so it should satisfy equation (1) i.e.,

0 = ¾ * (-20) + c $\Rightarrow$ c = 15 -----------------(2)

Using (2) in (1) we get the complete equation of line as

y = 3/4x + 15 ---------------------(3)

Steps to clip the line segment PQ are:

1) Point P is lying as LHS of window ABCD so by finding the intersection of line PQ with $y = y_{max} = 1$ we will get co-ordinates of P' using equation (3) put $y = y_{max} = 1$ in (3) we get,

   $1 = 3/4x + 15 \Rightarrow x = \frac{-14 * 4}{3} = -18.66$

   i.e., P'(-18.66, 1), which is lying outside the window region ABCDhence reject portion PP' of line segment PQ.

2) Now consider altered line segment P'Q

   By finding the pt. of intersection of P'Q with $x = x_{min} = -1$ we will get p'', so put $x = x_{min} = -1$ in (3) we get

   Y = ¾(-1) + 15 = -3/4 + 15 = 57/4 = 14.25

   $\therefore \Rightarrow$ P''(-1, 14.25) is also lying out of window region ABCD so reject portion P'P'' of line segment.

3) Now consider line segment P''Q

   We will get P''' if we will find the pt. of intersection of P''Q with $x = x_{max} = 3$

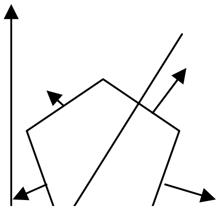
   $\therefore$by using $x = x_{max} = 3$ in (3) we get

   Y = ¾ * 3 + 15 = 9/4 + 15 = 69/4 = 17.25

   $\therefore \Rightarrow$ P'''(3, 17.25) is also lying out of window region ABCD hence Q (20,30) is also lying out $\therefore$Reject whole line segment P'''Q.

   That is how a whole line segment PQ is rejected in small segments.

2) The clipping window is given by $P_0(10,10)$, $P_1(20,10)$, $P_2(25,15)$, $P_3(20,20)$, $P_4(10,15)$, $P_5 = P_0$ and we need to clip the line A(0,0) and B(30,25). Using Cyrus Beck algorithm, we proceed as follows:



Steps to find clipping are:

1) Find $\overline{N}$ and using that find t:

$$t = \frac{-(\overline{P} - \overline{P_{i-1}}).\overline{N}}{(\overline{Q} - \overline{P}).\overline{N}}$$, $\overline{P}, \overline{Q}$ end pts. of line segment $\overline{P_{i-1}}$ starting pt of edge check t lies in the interval (0,1)

2) Find pt. of intersection $\overline{P} + t(\overline{Q} - \overline{P})$

3) If $(\overline{Q} - \overline{P}).\overline{N} < 0$ then pt. of intersection is potentially entering (P_E) if $(\overline{Q} - \overline{P}).\overline{N} > 0$ then pt. of intersections potentially leaving (P_L)

4) Find (max) t_{max} out of the value of t_{max} for P_E's and find(min) t_{mix} out of the value of t_{min} for P_L's condition of clip is $t_{max} < t_{min}$

5) Clipping is available from $\overline{P} + t_{max}(\overline{Q} - \overline{P})$ to $\overline{P} + t_{min}(\overline{Q} - \overline{P})$.

Now, let us do the above steps so as to clip the line $\overline{AB}$ in clipping windows p_0p_1,p_2,p_3,p_4.

edge 0: (i.e. edge P_0P_1)

Say $\overline{N_0} = (n_{01}, n_{02})$ be normal to this edge $\overline{p_0 p_1}$. So $\overline{N_0}.\overline{p_0 p_1} = 0$

$$\overrightarrow{p_0 p_1} = \overline{p_1} - \overline{p_0} = ((20,10) - (10,10) = (10,0) \qquad \text{--------------(1)}$$

$$\overline{N_0}.\overline{p_0 p_1} = (n_{01}, n_{02}).(10,0) = 0$$

$$= 10n_{01} + 0n_{02} = 0 \quad \Rightarrow \quad n_{01} = \frac{-0}{10} n_{02}$$

If we take $n_{02} = -1$ then $n_{01} = 0$ so, $\overline{N_0} = (0,1)$ --------------(2)

Now, to check that the point of intersection of line segment $\overline{AB}$ and edge is potentially entering or leaving find $\overline{AB}.\overline{N}$

$$\overline{AB} = \overline{B} - \overline{A} = ((30-0),(25-0)) = (30,25)$$

$$\overline{AB}.\overline{N_0} = (30,25).(0,1) = -25 < 0 \qquad so\ pt.us\ PE$$

$$t_{max} = \frac{-(\overline{A} - \overline{P_0}).\overline{N_0}}{(\overline{B} - \overline{A}).N_0} = \frac{-((0,0)-(10,10)).(0,1)}{-25} = \frac{(10,10).(0,1)}{-25} = \frac{2}{5}$$

edge 1: (i.e. edge $\overline{P_1 P_2}$)

Say, $\overline{N_1} = (n_{11}, n_{12})$ be normal to this edge $\overline{p_1 p_2}$. So $\overline{N_1}.\overline{p_1 p_2} = 0$

$$\overrightarrow{p_1 p_2} = \overline{p_2} - \overline{p_1} = ((25,20) - (15,10) = (5,5) \qquad \text{--------(1)}$$

$$(n_{11}, n_{12}).\overline{p_1 p_2} = (n_{11}, n_{12}).(5,5) = 5n_{11} + 5n_{12} = 0 \quad \Rightarrow \quad n_{11} = -n_{12}$$

If $n_{12} = -1$ then $n_{11} = 1$ $\therefore \overline{N_1} = (n_{11}, n_{12}) = (1,-1)$ -------------(2)

Now let's find $\overline{AB}.\overline{N_1}$ to check that point of intersection is potentially entering or leaving

$\overline{AB}.\overline{N_1}$ = (30,25).(1,-1) = 30-25 = 5 > 0 so pt. is P_L

So, find t_{min};

$$t_{\min} = \frac{-(\overline{A}-\overline{P_1}).\overline{N_1}}{(\overline{B}-\overline{A}).N_1} = \frac{-((0,0)-(20,10)).(1,-1)}{5} = \frac{(20,10).(1,-1)}{5} = \frac{20-10}{5} = 2;$$

reject this point of intersection

Edge 2: (i.e. edge $\overline{P_2P_3}$)

Say $\overline{N_2} = (n_{21}, n_{22})$ be normal to this edge $\overline{p_2p_3}$. So $\overline{N_2}.\overline{p_2p_3} = 0$

$$\overrightarrow{p_2p_3} = \overline{p_3} - \overline{p_2} = ((20-25),(20-15) = (-5,5) \qquad \text{--------(1)}$$

$$\overline{p_1p_2}.\overline{N_2} = (-5,5).(n_{21}, n_{22}) = 0 \Rightarrow -5n_{21} + 5n_{22} = 0 \Rightarrow n_{21} = n_{22}$$

If $n_{21} = 1$ then $n_{22} = 1$ $\therefore \overline{N_2} = (n_{21}, n_{22}) = (1,1)$ -------------(2)

To check that pt. of intersection in P_L or P_E, angle sign of $\overline{AB}.\overline{N_2}$

$\overline{AB}.\overline{N_2}$ = (30,25).(1,1) = 30+25 > 0 so P_L

So, find t_{min};

$$t_{\min} = \frac{-(\overline{A}-\overline{P_2}).\overline{N_2}}{(\overline{B}-\overline{A}).N_2} = \frac{-((0,0)-(25,15)).(1,1)}{55} = \frac{(25,15).(1,1)}{55} = \frac{25-15}{55} = \frac{40}{55} = \frac{8}{11}$$

Edge 3: (i.e. edge $\overline{P_3P_4}$)

Say $\overline{N_3}$ be normal to $\overline{p_3p_4}$, then $\overline{N_3}.\overline{p_3p_4} = 0$

$$\overrightarrow{p_3p_4} = \overline{p_4} - \overline{p_3} = ((10,15)-(20,20) = (-10,-5) \qquad \text{--------(1)}$$

$$\overline{p_1p_2}.\overline{N_3} = (-10,-5)(n_{31}, n_{32}) = 10n_{31} + 5n_{32} = 0 \quad \Rightarrow \quad n_{31} = -1/2\, n_{32}$$

if $n_{32} = 1$ then $n_{31} = -1/2$ $\therefore \overline{N_3} = (-1/2,-1)$ -------------(2)

Now, let us find that pt. of intersection is P_E or P_L, lets check sign of $\overline{AB}.\overline{N_3}$

$\overline{AB}.\overline{N_3}$ = (30,25).(-1/2,-1) = 30(-1/2)+25(1) = -15+25 =10 > 0 so P_L

$$t_{\min} = \frac{-(\overline{A}-\overline{P_3}).\overline{N_3}}{(\overline{B}-\overline{A}).N_3} = \frac{-((0,0)-(20,20)).(-1/2,1)}{10} = \frac{(20,20).(-1/2,1)}{10} = \frac{-10+20}{10} = 1$$

Edge 4: (i.e. edge $\overline{P_4P_5}$ or $\overline{P_4P_0}$ ($\therefore$ $P_5 = P_0$)

Say $\overline{N_4}(n_{41}, n_{42})$ be normal to the edge $\overline{p_4p_0}$, so $\overline{N_4}.\overline{p_4p_0} = 0$

$$\overrightarrow{p_4p_0} = \overline{p_0} - \overline{p_4} = ((10,10)-(10,15) = (0,-5) \qquad \text{--------(1)}$$

$$\overline{p_1p_2}.\overline{N_4} = (0,-5)(n_{41}, n_{42}) = 0n_{41} - 5n_{42} = 0 \quad \Rightarrow \quad n_{42} = \frac{0n_{41}}{5}$$

If $n_{41} = -1$ then $n_{42} = 0$ $\therefore \overline{N_4} = (-1,0)$ -------------(2)

Now to find whether that point of intersection is potentially entering or leaving find and check sign of $\overline{AB}.\overline{N_4}$

$\overline{AB}.\overline{N_4}$ = (30,25).(-1,0) = -30+0 < 0 so P_E

Now find that (t_{max});

$$t_{\max} = \frac{-(\overline{A}-\overline{P_4}).\overline{N_4}}{(\overline{B}-\overline{A}).N_4} = \frac{-((0,0)-(10,15)).(-1,0)}{-30} = \frac{(10,15).(-1,0)}{-30} = \frac{-10}{-30} = \frac{1}{3}$$

Now out of all t_{max} find max. value of t_{max} and
out of all t_{min} find min. value of t_{min} and
$max[t_{max}] = 2/5 = t_{max}$, $min[t_{min}] = 8/11 = t_{min}$.

as $[t_{max}] < [t_{min}]$. So the line segment AB is clipped from

$A + t_{max}(B-A)$ to $A + t_{min}(B-A)$

$\overline{A} + t_{max}(\overline{B} - \overline{A})$

$= (0,0) + 2/5[(30,25) - (0,0)] = (12, 10)$

$\overline{A} + t_{min}(\overline{B} - \overline{A})$

$= (0,0) + 8/11[(30,25) - (0,0)] = (240/11, 200/11)$

i.e., line is dipped from (12,10) to (240/11, 200/11), means

line AB is visible from (12,10) to (240/11, 200/11)

3) Let us compute the outward normal to a clipping in the case of Cyrus-Back algorithm. Consider a convex window, say it is hexagonal in shape; through which a line AB is passing and say $\overline{N_i}$ is normal at the edge of the hexagon where the line AB intersects i.e.,

$\overrightarrow{P_{i-1}P_i} = \overline{P_i} - \overline{P_{i-1}}$ is the edge with which the line segment AB intersects so,

$\overrightarrow{P_{i-1}P_i} = \overline{P_i} - \overline{P_{i-1}} = (x_i, y_i) - (x_{i-1}, y_{i-1})$ ----------------(1)

$= (x_i - x_{i-1}, y_i - y_{i-1}) = (s_1, s_2)$

If normal $\overline{N_i} = (n_{1i}, n_{2i})$ -------------------(2)

then, $\overrightarrow{P_{i-1}P_i}.\overrightarrow{N_i} = 0$

$(s_1, s_2).(n_{1i}, n_{2i}) = (s_1 n_{1i} + s_2 n_{2i}) = 0$

$n_{1i} = \frac{-s_2}{s_1} n_{2i}$

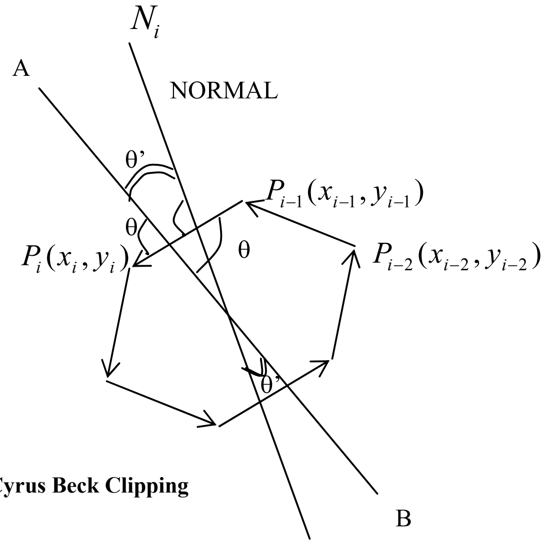


Figure 18: Example Cyrus Beck Clipping

Case1: If $n_{2i} = 1$ then $n_{1i} = \frac{-s_2}{s_1}$ $\therefore \overline{N_i} = (-s_2/s_1, 1)$

Case2: If $n_{2i} = -1$ then $n_{1i} = \frac{s_2}{s_1}$ $\therefore \overline{N_i} = (s_2/s_1, -1)$

Normal $\overline{N_i}$ in case 1 is opposite of the Normal in case 2 i.e. the direction is opposite.

So, if $\overline{N_i}$ in case is OUTWARD NORMAL then $\overline{N_i}$ in case2 will be INWARD NORMAL

In CYRUS-BECK Algorithm the point of intersection is either potentially entering on potentially leaving, lets study the two with inward/outward normal.

Case A: POTENTIALLY ENTERING CASE (*Figure 19*)
In *Figure19* the point of intersection given by $\overline{AB}$ is potentially entering

i) $\overline{N_i}$ is outward normal then,

$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos(\theta + 90°) = -|\overline{AB}||\overline{N_i}|\sin\theta < 0$$

ii) $\overline{N_i}$ is inward normal then,

$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos\theta' = |\overline{AB}||\overline{N_i}|\cos(90° - \theta) = -|\overline{AB}||\overline{N_i}|\sin\theta$$

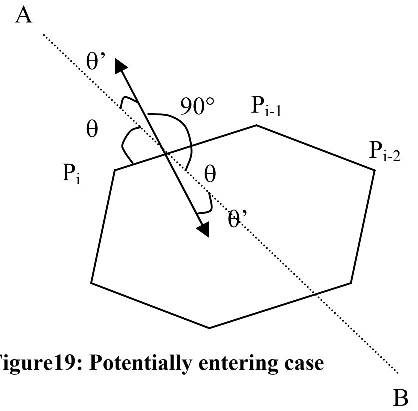


Figure19: Potentially entering case

Similarly ;

Case B: POTENTIALLY LEAVING CASE
When the point of intersection of line AB with edge $\overrightarrow{P_{i-1}.P_i}$ is potentially leaving.

i) $\overline{N_i}$ is outward normal then,

$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos\theta' = |\overline{AB}||\overline{N_i}|\cos(90° - \theta) = |\overline{AB}||\overline{N_i}|\sin\theta > 0$$

ii) $\overline{N_i}$ is inward normal then,

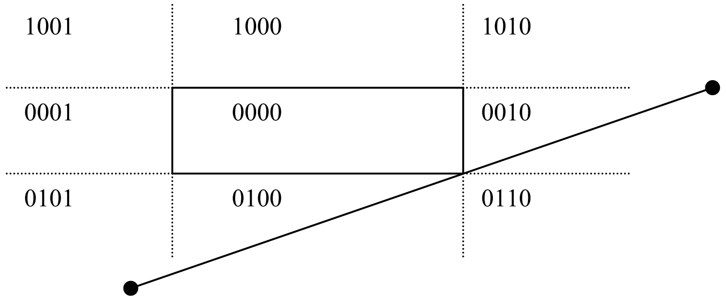
$$\overline{AB}.\overline{N_i} = |\overline{AB}||\overline{N_i}|\cos(90° + \theta) = -|\overline{AB}||\overline{N_i}|\sin\theta < 0$$

i.e., by analyzing the sign of dot product of $\overline{AB}$ with $\overline{N_i}$ in case of potentially entering/leaving pt. we can find that normal $\overline{N_i}$ is outward normal or inward normal.

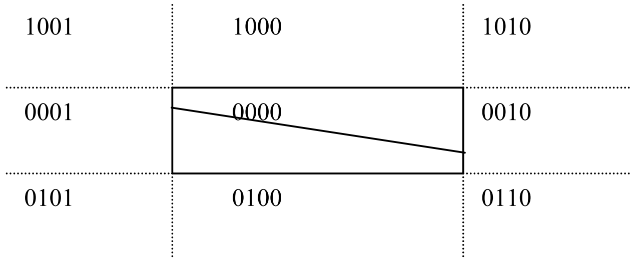
4) Limitations of Cohen-Sutherland clipping algorithm are:

1) Clipping window region can be rectangular in shape only and no other polygonal shaped window is allowed.
2) Edges of rectangular shaped clipping window has to be parallel to the x-axis and y-axis.
3) If end pts of line segment lies in the extreme limits i.e., one at R.H.S other at L.H.S., and on one the at top and other at the bottom (diagonally) then, even

if the line doesn't pass through the clipping region it will have logical intersection of 0000 implying that line segment will be clipped but infact it is not so.



5) Using Cohen Sutherland Line clipping algorithm when we clip the line we get the line shown in *Figure* below as output.



6) Window regions which are Non convex can't be handled by the CYRUS-BECK ALGORITHM without modification.

   As we know that in Cyrus Beck algorithm the line is to draw if tmax < tmin else it is to be rejected and as per equation (1) and Figure we find that the whole line LM will be rejected but some portion of LM has to be visible, to achieve this, the algorithm has to be modified so as to handle new convex window regions.

   Modifications to be applied to Cyrus Beck algorithm to handle the non convex region case are:

   1) The non convex region is to be divided into many convex sub regions.

   2) tmax and tmin are to be found w.r.t individual convex window region hence line is to be clipped for different convex window regions.

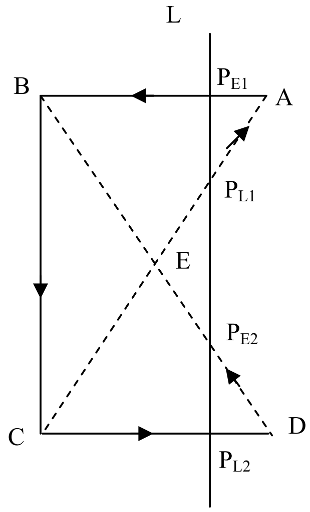
As per the modification the window region ABCDE is to be divided into two convex windows.

AEBA and BCDEB

Or

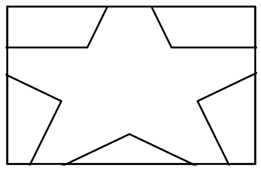
CDEC and ABCEA

ABCDE → Non convex window3
LM → line to be clipped
P_E → Potentially entering
P_L → Potentially leaving



Check Your Progress 2

1) Clipping the given polygon using Sutherland Hodgeman polygon clipping algorithm we will get the result given in the *Figure* below as output.



UNIT 1 2-D and 3-D TRANSFORMATIONS

Structure **Page Nos.**

1.0 Introduction 5
1.1 Objectives 5
1.2 Basic Transformations 6
1.2.1 Translation 6
1.2.2 Rotation 9
1.2.3 Scaling 12
1.2.4 Shearing 15
1.3 Composite Transformations 20
1.3.1 Rotation about a Point 21
1.3.2 Reflection about a Line 22
1.4 Homogeneous Coordinate Systems 27
1.5 3-D Transformations 31
1.5.1 Transformation for 3-1 Translations 32
1.5.2 Transformation for 3-D Rotation 32
1.5.3 Transformation for 3-D Scaling 34
1.5.4 Transformation for 3-D Shearing 35
1.5.5 Transformation for 3-D Reflection 35
1.6 Summary 37
1.7 Solutions / Answers 38

1.0 INTRODUCTION

In the previous Block, we have presented approaches for the generation of lines and polygonal regions. We know that once the objects are created, the different applications may require variations in these. For example, suppose we have created the scene of a room. As we move along the room we find the object's position comes closer to us, it appears bigger even as its orientation changes. Thus we need to alter or manipulate these objects. Essentially this process is carried out by means of transformations. Transformation is a process of changing the position of the object or maybe any combination of these.

The objects are referenced by their coordinates. Changes in orientation, size and shape are accomplished with **geometric transformations** that allow us to calculate the new coordinates. The basic geometric transformations are translation, rotation, scaling and shearing. The other transformations that are often applied to objects include reflection.

In this Block, we will present transformations to manipulate these geometric 2-D objects through Translation, and Rotation on the screen. We may like to modify their shapes either by magnifying or reducing their sizes by means of Scaling transformation. We can also find similar but new shapes by taking mirror reflection with respect to a chosen axis of references. Finally, we extend the 2-D transformations to 3-D cases.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- describe the basic transformations for 2-D translation, rotation, scaling and shearing;
- discuss the role of composite transformations;

- describe composite transformations for Rotation about a point and reflection about a line;
- define and explain the use of homogeneous coordinate systems for the transformations, and
- extend the 2-D transformations discussed in the unit to 3-D transformations.

1.2 BASIC TRANSFORMATIONS

Consider the xy-coordinate system on a plane. An object (say *Obj*) in a plane can be considered as a set of points. Every object point P has coordinates (x,y), so the object is the sum total of all its coordinate points (see *Figure 1*). Let the object be moved to a new position. All the coordinate points P'(x',y') of a new object *Obj'* can be obtained from the original points P(x,y) by the application of a geometric transformation.

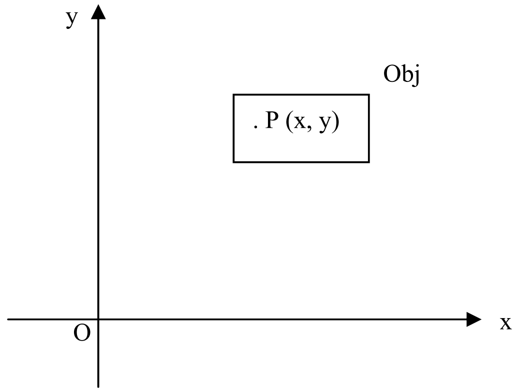


Figure 1

1.2.1 Translation

Translation is the process of changing the position of an object. Let an object point P(x,y)=xI+yJ be moved to P'(x',y') by the given translation vector V= t_xI + t_yJ, where t_x and t_y is the translation factor in x and y directions, such that

P'=P+V. ----------------(1)

In component form, we have

$$T_v = \begin{cases} x' = x + t_x \text{ and} \\ y' = y + t_y \end{cases} \qquad \text{----------------(2)}$$

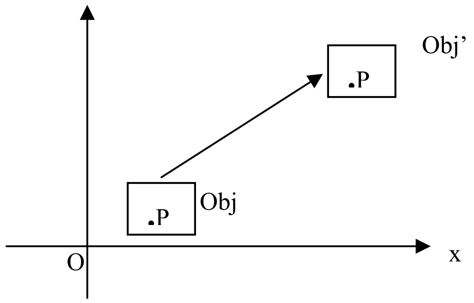


Figure 2

As shown in *Figure 2*, P' is the new location of P, after moving t_x along x-axis and t_y along y-axis. It is not possible to develop a relation of the form.

$P' = P.T_v$ ---------------(3)

Where Tv is the transformation for translation **in matrix form**.

That is, we cannot represent the translation transformation in (2x2) matrix form (2-D Euclidean system).

Any transformation operation can be represented as a (2x2) matrix form, except translation, i.e., translation transformation cannot be expressed as a (2x2) matrix form (2-D Euclidean system). But by using Homogeneous coordinate system (HCS), we can represent translation transformation in matrix form. The HCS and advantages of using HCS is discussed, in detail, in section 1.4.

Relation between 2-D Euclidean (Cartesian) system and HCS

Let P(x,y) be any point in 2-D Euclidean system. In Homogeneous Coordinate system, we add a third coordinate to the point. Instead of (x,y), each point is represented by a triple (x,y,H) such that H≠0; with the condition that (x1,y1,H1)=(x2,y2,H2) $\leftrightarrow$ x1/H1 = x2/H2 ; y1/H1 = y2/H2. In two dimensions the value of H is usually kept at 1 for simplicity. (If we take H=0 here, then this represents point at infinity, i.e, generation of horizons).

The following table shows a relationship between 2-D Euclidean (Cartesian coordinate) system and HCS.

2-D Euclidian System	Homogeneous Coordinate System (HCS)
Any point (x,y) $\longrightarrow$	(x,y,1)
H≠0); (x/H,y/H) $\longleftarrow$	If (x,y,H) be any point in HCS(such that then (x,y,H)=(x/H,y/H,1), i.e. (x,y,H)

For translation transformation, any point (x,y)→(x+tx,y+ty) in 2-D Euclidian system. Using HCS, this translation transformation can be represented as (x,y,1) → (x+tx,y+ty,1). In two dimensions the value of H is usually kept at 1 for simplicity. Now, we are able to represent this translation transformation in matrix form as:

$$(x',y',1)=(x,y,1)\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{pmatrix}$$

$P'_h = P_h.T_v$ ----------------(4)

Where P'_h and P_h represents object points in Homogeneous Coordinates and Tv is called homogeneous transformation matrix for translation. Thus, P'_h, the new coordinates of a transformed object, can be found by multiplying previous object coordinate matrix, P_h, with the transformation matrix for translation Tv.

The ***advantage*** of introducing the matrix form of translation is that it simplifies the operations on complex objects i.e., we can now build complex transformations by multiplying the basic matrix transformations. This process is called *concatenation of*

matrices and the resulting matrix is often referred as the *composite transformation* matrix.

We can represent the basic transformations such as rotation, scaling shearing, etc., as 3x3 homogeneous coordinate matrices to make matrix multiplication compatibility with the matrix of translation. This is accomplished by augmenting the 2x2 matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with a third column $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ and a third row (0,0,1). That is

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thus, the new coordinates of a transformed object can be found by multiplying previous object coordinate matrix with the required transformation matrix. That is

$$\begin{bmatrix} \text{New Object} \\ \text{Coordinate} \\ \text{matrix} \end{bmatrix} = \begin{bmatrix} \text{Previous object} \\ \text{Coordinate} \\ \text{matrix} \end{bmatrix} . \begin{bmatrix} \text{Transformation} \\ \text{matrix} \end{bmatrix}$$

Example1: Translate a square ABCD with the coordinates A(0,0),B(5,0),C(5,5),D(0,5) by 2 units in x-direction and 3 units in y-direction.

Solution: We can represent the given square, in matrix form, using homogeneous coordinates of vertices as:

$$\begin{matrix} A \\ B \\ C \\ D \end{matrix} \begin{pmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \\ x4 & y4 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 5 & 0 & 1 \\ 5 & 5 & 1 \\ 0 & 5 & 1 \end{pmatrix}$$

The translation factors are, tx=2, ty=3

The transformation matrix for translation :

$$T_v = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 3 & 1 \end{bmatrix}$$

New object point coordinates are:

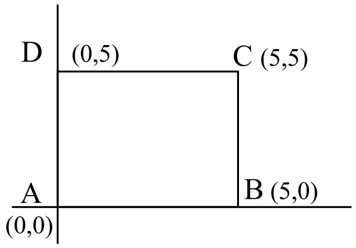
$$[A'B'C'D'] = [ABCD].T_v$$

$$\begin{matrix} A' \\ B' \\ C' \\ D' \end{matrix} \begin{pmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ x'_3 & y'_3 & 1 \\ x'_4 & y'_4 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 5 & 0 & 1 \\ 5 & 5 & 1 \\ 0 & 5 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 3 & 1 \end{pmatrix}$$

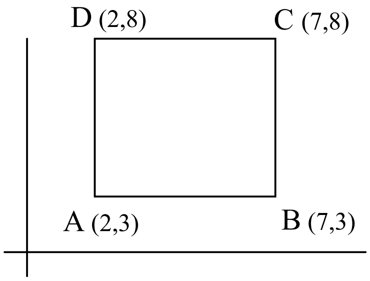
$$= \begin{pmatrix} 2 & 3 & 1 \\ 7 & 3 & 1 \\ 7 & 8 & 1 \\ 2 & 8 & 1 \end{pmatrix}$$

Thus, $A'(x'_1,y'_1)=(2,3)$
$B'(x'_2,y'_2)=(7,3)$
$C'(x'_3,y'_3)=(7,8)$ and $D'(x'_4,y'_4)=(2,8)$

The graphical representation is given below:



a) Square before Translation



b) Square after Translation

1.2.2 Rotation

In 2-D rotation, an object is rotated by an angle θ with respect to the origin. This angle is assumed to be positive for anticlockwise rotation. There are two cases for 2-D rotation, *case1*- rotation about the origin and *case2* rotation about an arbitrary point. If, the rotation is made about an arbitrary point, a set of basic transformation, i.e., composite transformation is required. For 3-D rotation involving 3-D objects, we need to specify both the angle of rotation and the axis of rotation, about which rotation has to be made. Here, we will consider case1 and in the next section we will consider *case2.*

Before starting case-1 or case-2 you must know the relationship between **polar coordinate system** and **Cartesian system:**

Relation between **polar coordinate system** and **Cartesian system**

A frequently used non-cartesian system is Polar coordinate system. The following *Figure A* shows a polar coordinate reference frame. In polar coordinate system a coordinate position is specified by r and θ, where r is a radial distance from the coordinate origin and θ is an angular displacements from the horizontal (see *Figure 2A*). Positive angular displacements are counter clockwise. Angle θ is measured in degrees. One complete counter-clockwise revolution about the origin is treated as 360^0. A relation between Cartesian and polar coordinate system is shown in *Figure 2B*.

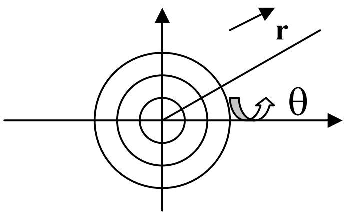


Figure 2A: A polar coordinate reference-frame

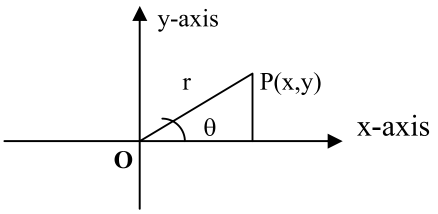


Figure 2B: Relation between Polar and Cartesian coordinates

Consider a right angle triangle in *Figure B*. Using the definition of trigonometric functions, we transform polar coordinates to Cartesian coordinates as:

$x=r.\cos\theta$
$y=r.\sin\theta$

The inverse transformation from Cartesian to Polar coordinates is:

$r=\sqrt{(x^2+y^2)}$ and $\theta=\tan^{-1}(y/x)$

Case 1: Rotation about the origin

Given a 2-D point P(x,y), which we want to rotate, with respect to the origin O. The vector OP has a length 'r' and making a positive (anticlockwise) angle ϕ with respect to x-axis.

Let P'(x'y') be the result of rotation of point P by an angle θ about the origin, which is shown in *Figure 3*.

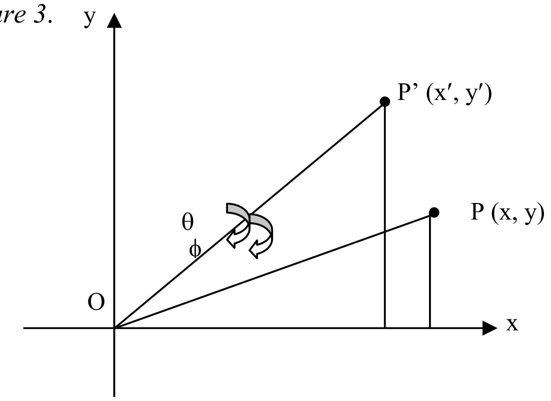


Figure 3

$P(x,y) = P(r.\cos\phi, r.\sin\phi)$
$P'(x',y') = P[r.\cos(\phi+\theta), r\sin(\phi+\theta)]$

The coordinates of P' are:

$x' = r.\cos(\theta+\phi) = r(\cos\theta\cos\phi - \sin\theta\sin\phi)$
$= x.\cos\theta - y.\sin\theta$ (where $x = r\cos\phi$ and $y = r\sin\phi$)

similarly;

$y' = r\sin(\theta+\phi) = r(\sin\theta\cos\phi + \cos\theta.\sin\phi)$
$= x\sin\theta + y\cos\theta$

Thus,

$$R_\theta = \left\{ \begin{array}{l} x' = x.\cos\theta - y.\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{array} \right\} = R_\theta$$

Thus, we have obtained the new coordinate of point P after the rotation. In matrix form, the transformation relation between P' and P is given by:

$$(x'y') = (x,y) \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

that is $P' = P.R_\theta$ ---------(5)

where P'and P represent object points in 2-D Euclidean system and R_θ is transformation matrix for **anti-clockwise** Rotation.

In terms of HCS, equation (5) becomes

$$(x', y', 1) = (x, y, 1) \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{---------(6)}$$

That is $P'_h = P_h.R_\theta$, ---------(7)

Where P'_h and P_h represents object points, after and before required transformation, in Homogeneous Coordinates and R_θ is called homogeneous transformation matrix for **anticlockwise** Rotation. Thus, P'_h, the new coordinates of a transformed object, can be found by multiplying previous object coordinate matrix, P_h, with the transformation matrix for Rotation R_θ.

Note that for **clockwise** rotation we have to put θ = – θ, thus the rotation matrix R_θ, in HCS, becomes

$$R_{-\theta} = \begin{pmatrix} \cos(-\theta) & \sin(-\theta) & 0 \\ -\sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Example 2: Perform a 45^0 rotation of a triangle A(0,0),B(1,1),C(5,2) about the origin.

Solution: We can represent the given triangle, in matrix form, using homogeneous coordinates of the vertices:

$$[ABC] = \begin{pmatrix} A & 0 & 0 & 1 \\ B & 1 & 1 & 1 \\ C & 5 & 2 & 1 \end{pmatrix}$$

The matrix of rotation is: $R_\theta = R_{45^0} = \begin{pmatrix} \cos45^0 & \sin45^0 & 0 \\ -\sin45^0 & \cos45^0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

So the new coordinates A'B'C' of the rotated triangle ABC can be found as:

$$[A'B'C'] = [ABC].\,R_{45^\circ} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \sqrt{2} & 1 \\ 3\sqrt{2}/2 & 7\sqrt{2}/2 & 1 \end{bmatrix}$$

Thus A'=(0,0), B'=(0,√2), C'=(3√2/2,7√2/2)

The following *Figure (a)* shows the original, triangle [ABC] and *Figure (b)* shows triangle after the rotation.

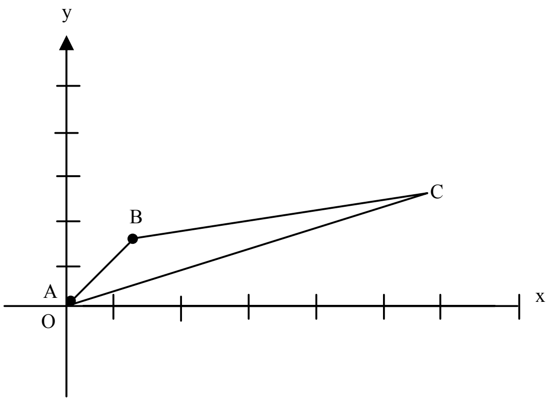


Figure (a)

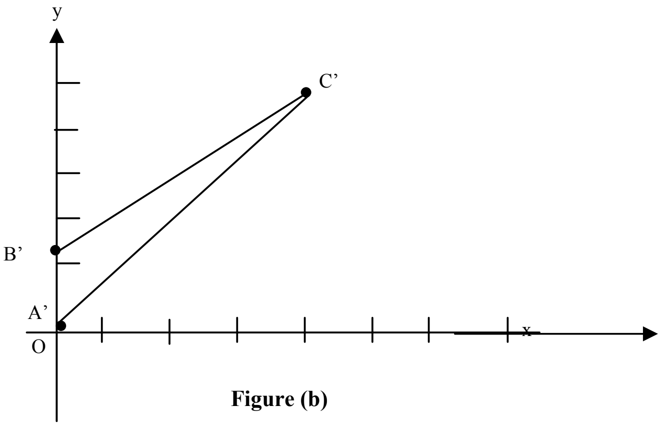


Figure (b)

☞ Check Your Progress 1

1) What are the basic advantages of using Homogeneous coordinates system.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) A square consists of vertices A(0,0),B(0,1),C(1,1),D(1,0). After the translation C is found to be at the new location (6,7). Determine the new location of other vertices.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) A point P(3,3) makes a rotating of 45^0 about the origin and then translating in the direction of vector v=5I+6J. Find the new location of P.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

4) Find the relationship between the rotations R_θ, $R_{-\theta}$, and R_θ^{-1} .

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

1.2.3 Scaling

Scaling is the process of expanding or compressing the dimensions (i.e., size) of an object. An important application of scaling is in the development of viewing transformation, which is a mapping from a window used to clip the scene to a view port for displaying the clipped scene on the screen.

Let P(x,y) be any point of a given object and s_x and s_y be scaling factors in x and y directions respectively, then the coordinate of the scaled object can be obtained as:

$$\left.\begin{array}{l} x' = x.s_x \\ y' = y.s_y \end{array}\right\} \qquad \text{--------(8)}$$

If the scale factor is 0<s<1, then it reduces the size of an object and if it is more then 1, it magnifies the size of the object along an axis.

For example, assume s_x >1.

i) Consider (x,y)→(2.x,y) i.e., Magnification in x-direction with scale factor s_x =2.

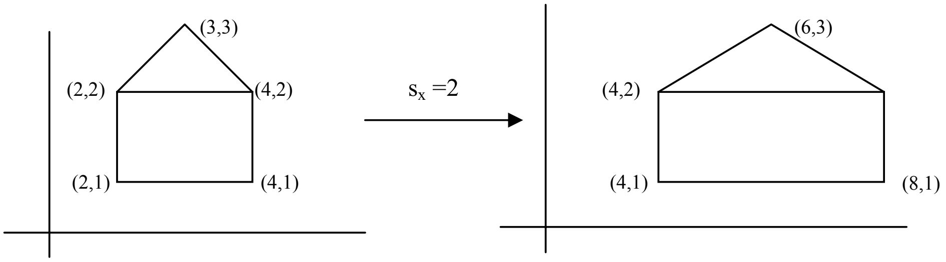


Figure a): Object before Scaling **Figure b): Object after Scaling with s_x =2**

ii) Similarly, assume s_y >1 and consider (x,y)→(x,2.y), i.e., Magnification in y-direction with scale factor s_y =2.

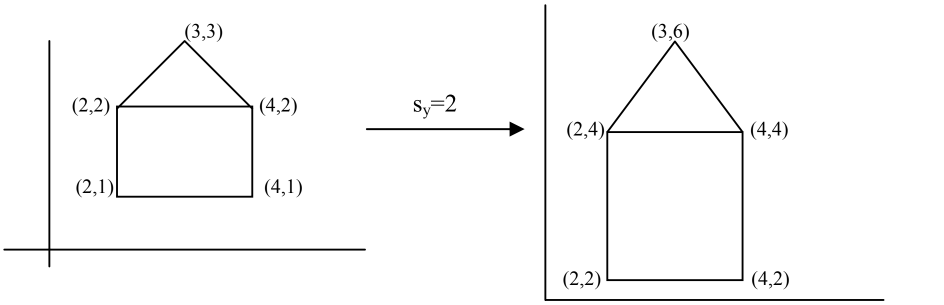


Figure a): Object before Scaling **Figure b): Object after Scaling with Sy=2**

iii) Consider (x,y)→(x.s_x,y) where 0< s_x = y_2 <1 i.e., Compression in x-direction with scale factor s_x=1/2.

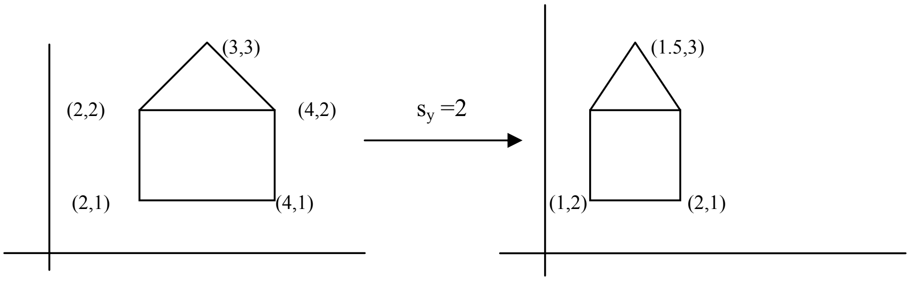


Figure a): Object before Scaling **Figure b): Object after Scaling with Sx=1/2**

Thus, the general scaling is (x,y)→ (x.s_x,y.s_y) i.e., magnifying or compression in both x and y directions depending on Scale factors s_x and s_y. We can represent this in matrix form (2-D Euclidean system) as:

$$(x',y') = (x,y)\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \qquad \text{----(9)}$$

In terms of HCS, equation (9) becomes:

$$(x',y',1)=(x,y,1)\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{----(10)}$$

that is $P'_h=P_h.s_{sx,sy}$ ----(11)

Where P_h and P'_h represents object points, before and after required transformation, in Homogeneous Coordinates and $s_{sx,sy}$ is called transformation matrix for general scaling with scaling factor s_x and s_y .

Thus, we have seen any positive value can be assigned to scale factors s_x and s_y. We have the following three cases for scaling:

Case 1: If the values of s_x and s_y are less then 1, then the size of the object will be reduced.

Case2: If both s_x and s_y are greater then 1, then the size of the object is enlarged.

Case3: If we have the same scaling factor (i.e. $s_x=s_x=S$), then there will be uniform scaling (either enlargement or compression depending on the value of S_x and Sy) in both x and y directions.

Example3: Find the new coordinates of a triangle A(0,0),B(1,1),C(5,2) after it has been (a) magnified to twice its size and (b) reduced to half its size.

Solution: Magnification and reduction can be achieved by a uniform scaling of s units in both the x and y directions. If, s>1, the scaling produces magnification. If, s<1, the result is a reduction. The transformation can be written as: (x,y,1)→(s.x,s.y,1). In matrix form this becomes

$$(x,y,1).\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} = (s.x,s.y,1)$$

We can represent the given triangle, shown in *Figure (a)*, in matrix form, using homogeneous coordinates of the vertices as :

$$\begin{bmatrix} A & 0 & 0 & 1 \\ B & 1 & 1 & 1 \\ C & 5 & 2 & 1 \end{bmatrix}$$

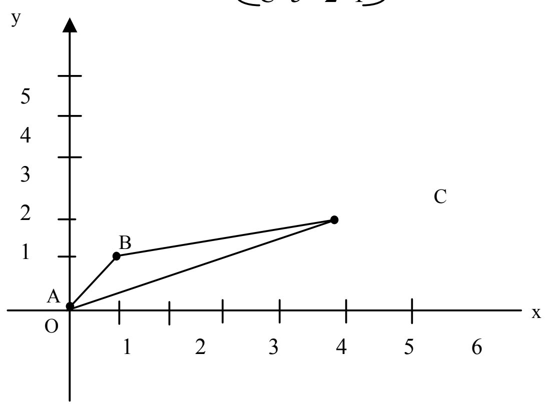


Figure a: Object before scaling

(a) choosing s=2

The matrix of scaling is: $S_{s_x,s_y} = S_{2,2} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

So the new coordinates A'B'C' of the scaled triangle ABC can be found as:

$$[A'B'C'] = [ABC].\ R_{2,2} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 2 & 1 \\ 10 & 4 & 1 \end{bmatrix}$$

Thus, A'=(0,0), B'=(2,2), C'= (10, 4)

(b) Similarly, here, s=1/2 and the new coordinates are A''=(0,0), B''=(1/2,1/2), C''=(5/2,1). The following figure (b) shows the effect of scaling with $s_x=s_y=2$ and (c) with $s_x=s_y=s=1/2$.

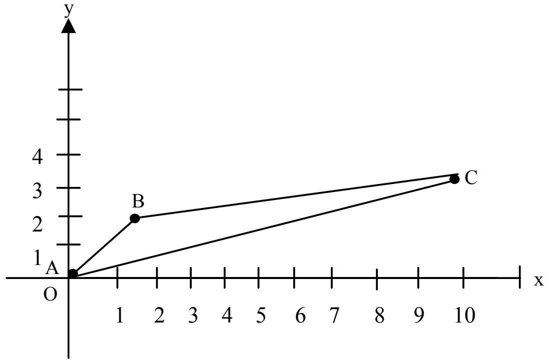


Figure b: Object after scaling with Sx = Sy = 2

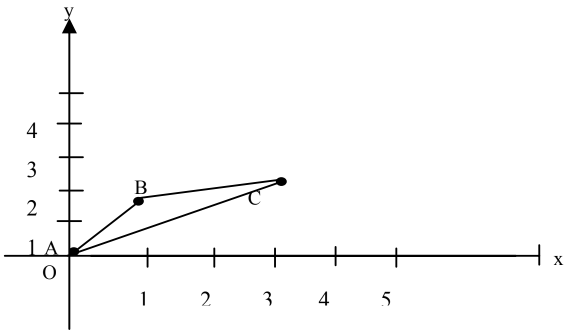


Figure c: Object after scaling with $S_x = S_y = 1/2$

1.2.4 Shearing

Shearing transformations are used for modifying the shapes of 2-D or 3-D objects. The effect of a shear transformation looks like "pushing" a geometric object in a direction that is parallel to a coordinate plane (3D) or a coordinate axis (2D). How far a direction is pushed is determined by its *shearing factor.*

One familiar example of shear is that observed when the top of a book is moved relative to the bottom which is fixed on the table.

In case of 2-D shearing, we have two types namely *x-shear* and *y-shear*.

In *x-shear*, one can push in the *x*-direction, positive or negative, and keep the *y*-direction unchanged, while in *y-shear*, one can push in the *y*-direction and keep the *x*-direction fixed.

x-shear about the origin

Let an object point P(x,y) be moved to P'(x',y') in the x-direction, by the given scale parameter 'a',i.e., P'(x'y') be the result of x-shear of point P(x,y) by scale factor a about the origin, which is shown in *Figure 4.*

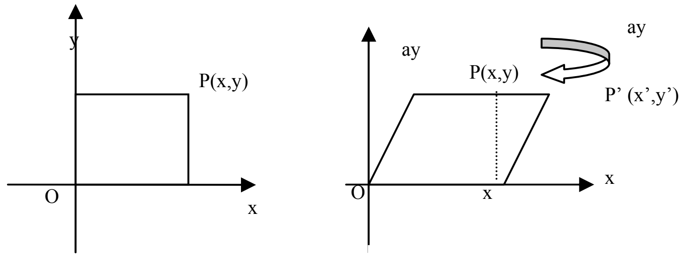


Figure 4

Thus, the points P(x,y) and P'(x',y') have the following relationship:

$$\left.\begin{array}{l} x' = x + ay \\ y' = y \end{array}\right\} = Sh_x(a) \qquad \text{------(11a)}$$

where 'a' is a constant (known as shear parameter) that measures the degree of shearing. If a is negative then the shearing is in the opposite direction.

Note that P(0,H) is taken into P'(aH,H). It follows that the shearing angle A (the angle through which the vertical edge was sheared) is given by:

tan(A) = aH/H = a.

So the parameter a is just the tan of the shearing angle. In matrix form (2-D Euclidean system), we have

$$(x',y')=(x,y)\begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \qquad \text{-------(12)}$$

In terms of Homogeneous Coordinates, equation (12) becomes

$$(x',y',1)=(x,y,1).\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{-------(13)}$$

That is, $P'_h = P_h\, Sh_x(a)$ --------(14)

Where P_h and P'_h represents object points, before and after required transformation, in Homogeneous Coordinates and $Sh_x(a)$ is called homogeneous transformation matrix for x-shear with scale parameter 'a' in the x-direction.

y-shear about the origin

Let an object point P(x,y) be moved to P'(x',y') in the x-direction, by the given scale parameter 'b'. i.e., P'(x'y') be the result of y-shear of point P(x,y) by scale factor 'b' about the origin, which is shown in *Figure 5(a)*.

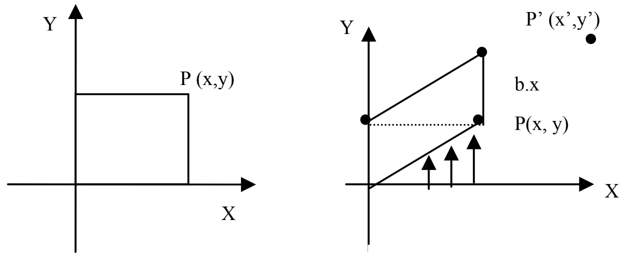


Figure 5 (a)

Thus, the points P(x,y) and P'(x',y') have the following relationship :

$$\left.\begin{aligned} x' &= x \\ y' &= y+bx \end{aligned}\right\} = Sh_y(b) \qquad \text{----------(15)}$$

where 'b' is a constant (known as shear parameter) that measures the degree of shearing. In matrix form, we have

$$(x',y')=(x,y)\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \qquad \text{----------(16)}$$

In terms of Homogeneous Coordinates, equation (16) becomes

$$(x',y',1)=(x,y,1)\begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{---------(17)}$$

That is, $P'_h = P_h.Sh_y(b)$ ---------(18)

Where P_h and P'_h represents object points, before and after required transformation, in Homogeneous Coordinates and $Sh_y(b)$ is called homogeneous transformation matrix for y-shear with scale factor 'b' in the y-direction.

xy-shear about the origin

Let an object point P(x,y) be moved to P'(x',y') as a result of shear transformation in both *x*- and *y*-directions with shearing factors *a* and *b*, respectively, as shown in *Figure 5(b)*.

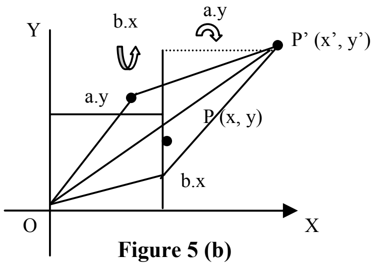


Figure 5 (b)

The points P(x,y) and P'(x',y') have the following relationship :

$$\left.\begin{aligned} x' &= x + ay \\ y' &= y+bx \end{aligned}\right\} = Sh_{xy}(a,b) \qquad \text{----------(19)}$$

where 'ay' and 'bx' are shear factors in x and y directions, respectively. The xy-shear is also called *simultaneous shearing* or *shearing* for short.

In matrix form, we have,

$$(x',y')=(x,y)\begin{bmatrix} 1 & b \\ a & 1 \end{bmatrix} \qquad \text{---------(20)}$$

In terms of Homogeneous Coordinates, we have

$$(x',y',1)=(x,y,1)\begin{bmatrix} 1 & b & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{---------(21)}$$

That is, $P'_h = P_h.Sh_{xy}(a,b)$ ----------(22)

Where P_h and P'_h represent object points, before and after required transformation, in Homogeneous Coordinates and $Sh_{xy}(a,b)$ is called homogeneous transformation matrix for xy-shear in both *x*- and *y*-directions with shearing factors *a* and *b*, respectively,

Special case: when we put b=0 in equation (21), we have *shearing in x-direction*, and when a=0, we have *Shearing in the y-direction,* respectively**.**

Example 4: A square ABCD is given with vertices A(0,0),B(1,0),C(1,1), and D(0,1). Illustrate the effect of a) x-shear b) y-shear c) xy-shear on the given square, when a=2 and b=3.

Solution: We can represent the given square ABCD, in matrix form, using homogeneous coordinates of vertices as:

$$\begin{pmatrix} A & 0 & 0 & 1 \\ B & 1 & 0 & 1 \\ C & 1 & 1 & 1 \\ D & 0 & 1 & 1 \end{pmatrix}$$

a) The matrix of x-shear is:

$$Sh_x(a) = \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

So the new coordinates A'B'C'D' of the x-sheared object ABCD can be found as:
[A'B'C'D']=[ABCD]. $Sh_x(a)$

$$[A'B'C'D'] = \begin{pmatrix} A & 0 & 0 & 1 \\ B & 1 & 0 & 1 \\ C & 1 & 1 & 1 \\ D & 0 & 1 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 3 & 1 & 1 \\ 2 & 1 & 1 \end{pmatrix}$$

Thus, A'=(0,0), B'=(1,0), C'=(3,1) and D'=(2,1).

b) Similarly the effect of shearing in the y direction can be found as:
[A'B'C'D']=[ABCD].$Sh_y(b)$

$$[A'B'C'D'] = \begin{pmatrix} A & 0 & 0 & 1 \\ B & 1 & 0 & 1 \\ C & 1 & 1 & 1 \\ D & 0 & 1 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 3 & 1 \\ 1 & 4 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Thus, A'=(0,0), B'=(1,3), C'=(1,4) and D'=(0,1).

c) Finally the effect of shearing in both directions can be found as:
[A'B'C'D']=[ABCD]. $Sh_{xy}(a,b)$

$$[A'B'C'D'] = \begin{pmatrix} A & 0 & 0 & 1 \\ B & 1 & 0 & 1 \\ C & 1 & 1 & 1 \\ D & 0 & 1 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 3 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 3 & 1 \\ 3 & 4 & 1 \\ 2 & 1 & 1 \end{pmatrix}$$

Thus, A'=(0,0), B'=(1,3), C'=(3,4) and D'=(2,1).

Figure (a) shows the original square, *figure (b)-(d)* shows shearing in the x, y and both directions respectively.

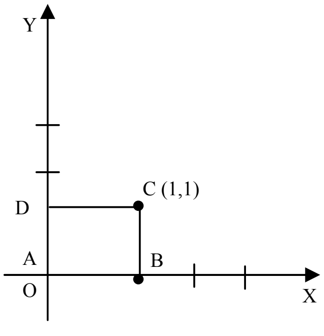


Figure (a)

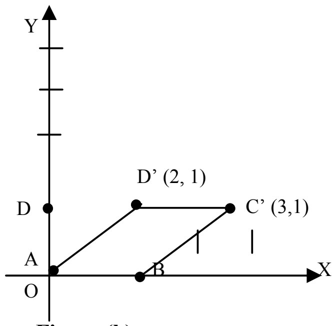


Figure (b)

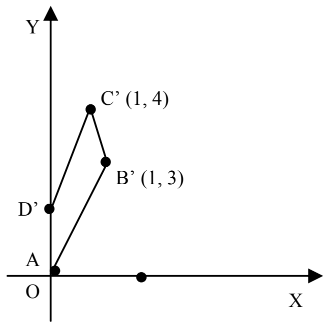


Figure (c)

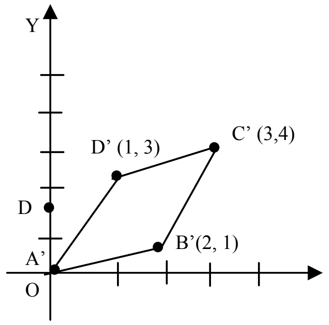


Figure (d)

Example 5: What is the use of Inverse transformation? Give the Inverse transformation for translation, rotation, reflection, scaling, and shearing.

Solution: We have seen the basic matrix transformations for translation, rotation, reflection, scaling and shearing with respect to the origin of the coordinate system. By multiplying these basic matrix transformations, we can build complex transformations, such as rotation about an arbitrary point, mirror reflection about a line etc. This process is called *concatenation of matrices* and the resulting matrix is often referred to as the *composite transformation matrix*. Inverse transformations play an important role when you are dealing with composite transformation. They come to the rescue of basic transformations by making them applicable during the construction of composite transformation. You can observed that the Inverse transformations for translation, rotation, reflection, scaling and shearing have the following relations, and v, θ, a, b, sx, sy, sz are all parameter involved in the transformations.

1) $T_v^{-1} = T_{-v}$
2) $R_\theta^{-1} = R_{-\theta}$
3) (i) $Sh_x^{-1}(a) = Sh_x(-a)$
   (ii) $Sh_y^{-1}(b) = Sh_x(-b)$
   (iii) $Sh_{xy}^{-1}(a,b) = Sh_x(-a,-b)$
4) $S^{-1}_{sx,sy,sz} = S_{1/sx,1/sy,1/sz}$
5) The transformation for mirror reflection about principal axes do not change after inversion.
   (i) $Mx^{-1} = M_{-x} = M_x$
   (ii) $My^{-1} = M_{-y} = M_y$
   (iii) $Mz^{-1} = M_{-z} = M_z$,

6) The transformation for rotations made about x,y,z axes have the following inverse:
(i) $R^{-1}_{x,\theta} = R_{x,-\theta} = R^{T}_{x,\theta}$
(ii) $R^{-1}_{y,\theta} = R_{y,-\theta} = R^{T}_{y,\theta}$
(iii) $R^{-1}_{z,\theta} = R_{z,-\theta} = R^{T}_{z,\theta}$

☞ Check Your Progress 2

1) Differentiate between the Scaling and Shearing transformation.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) Show that $S_{a,b} . S_{c,d} = S_{c,d} . S_{a,b} = S_{ac,bd}$

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) Find the 3x3 homogeneous co-ordinate transformation matrix for each of the following:
a) Shift an image to the right by 3 units.
b) Shift the image up by 2 units and down 1 units.
c) Move the image down 2/3 units and left 4 units.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

4) Find the condition under which we have $Ss_{x},s_{y} .R_{\theta} = R_{\theta}. Ss_{x},s_{y}$.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

5) Is a simultaneous shearing the same as the shearing in one direction followed by a shearing in another direction? Why?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

1.3 COMPOSITE TRANSFORMATIONS

We can build complex transformations such as rotation about an arbitrary point, mirror reflection about a line, etc., by multiplying the basic matrix transformations. This process is called *concatenation of matrices* and the resulting matrix is often referred to as the *composite transformation matrix*. In composite transformation, a previous transformation is pre-multiplied with the next one.

In other words we can say that a sequence of the transformation matrices can be concatenated into a single matrix. This is an effective procedure as it reduces because instead of applying initial coordinate position of an object to each transformation matrix, we can obtain the final transformed position of an object by applying composite matrix to the initial coordinate position of an object. In other words we can say that a sequence of transformation matrix can be concatenated matrix into a single matrix. This is an effective procedure as it reduces computation because instead of applying initial coordinate position of an object to each transformation matrix, we can obtain the final transformed position of an object by applying composite matrix to the initial coordinate position of an object.

1.3.1 Rotation about a Point

Given a 2-D point P(x,y), which we want to rotate, with respect to an arbitrary point A(h,k). Let P'(x'y') be the result of anticlockwise rotation of point P by angle θ about A, which is shown in *Figure 6*.

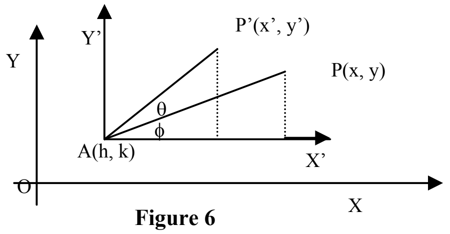


Figure 6

Since, the rotation matrix R_θ is defined only with respect to the origin, we need a set of basic transformations, which constitutes the composite transformation to compute the rotation about a given arbitrary point A, denoted by $R_{\theta,A}$. We can determine the transformation $R_{\theta,A}$ in three steps:

1) Translate the point A(h,k) to the origin O, so that the center of rotation A is at the origin.
2) Perform the required rotation of θ degrees about the origin, and
3) Translate the origin back to the original position A(h,k).

Using **v**=h**I**+k**J** as the translation vector, we have the following sequence of three transformations:

$$R_{\theta,A} = T_{-v} . R_\theta . T_v$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -h & -k & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ h & k & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ (1-\cos\theta).h+k.\sin\theta & (1-\cos\theta).k-h.\sin\theta & 1 \end{pmatrix} \quad \text{-------(23)}$$

Example 5: Perform a 45^0 rotation of a triangle A (0,0), B (1,1), C (5,2) about an arbitrary point P(–1, –1).

Solution: Given triangle ABC, as show in Figure (a), can be represented in homogeneous coordinates of vertices as:

$$[A\,B\,C] = \begin{matrix} A \\ B \\ C \end{matrix}\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{pmatrix}$$

From equation (23), a rotation matrix R_Q, A about a given arbitrary point A (h, k) is:

$$R_q, A = \begin{pmatrix} Cos\,\theta & Sin\,\theta & 0 \\ -Sin\,\theta & Cos\,\theta & 0 \\ (1-Cos\,\theta).h + k.Sin\,\theta & (1-Cos\,\theta).k - h.Sin\,\theta & 1 \end{pmatrix}$$

Thus $R_{45^o,}\, A = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -1 & (\sqrt{2}-1) & 1 \end{pmatrix}$

So the new coordinates $[A'\,B'\,C']$ of the rotated triangle [ABC] can be found as:

$$[A'\,B'\,C'] = [ABC]\,.\,R_{45^o,\,A} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{pmatrix} . \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ -1 & (\sqrt{2}-1) & 1 \end{pmatrix} =$$

$$\begin{matrix} A' \\ B' \\ C' \end{matrix}\begin{bmatrix} -1 & (\sqrt{2}-1) & 1 \\ -1 & 2\sqrt{2}-1 & 1 \\ \left(\frac{3}{2}\sqrt{2}-1\right) & \left(\frac{9}{2}.\sqrt{2}-1\right) & 1 \end{bmatrix}$$

Thus, $A' = (-1,\ \sqrt{2}-1)$, $B' = (-1,\ 2\sqrt{2}-1)$, and $C' = \left(\frac{3}{2}\sqrt{2}-1, \frac{9}{2}\sqrt{2}-1\right)$. The following *figure (a) and (b)* shows a given triangle, before and after the rotation.

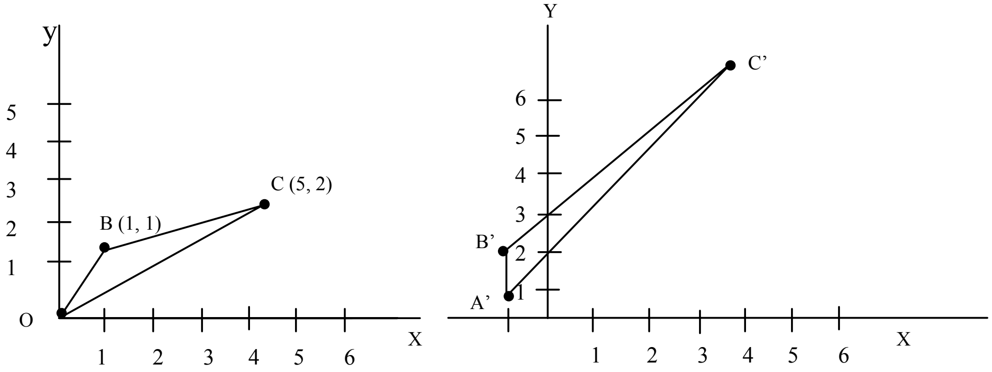


Figure (a) **Figure (b)**

1.3.2 Reflection about a Line

Reflection is a transformation which generates the mirror image of an object. As discussed in the previous block, the mirror reflection helps in achieving 8-way symmetry for the circle to simplify the scan conversion process. For reflection we need to know the reference axis or reference plane depending on whether the object is 2-D or 3-D.

Let the line L be represented by y=mx+c, where 'm' is the slope with respect to the x axis, and 'c' is the intercept on y-axis, as shown in *Figure 7*. Let P'(x',y') be the mirror reflection about the line L of point P(x,y).

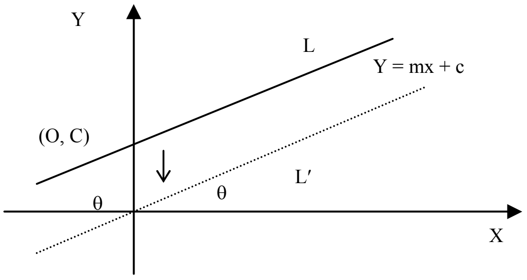


Figure 7

The transformation about mirror reflection about this line L consists of the following basic transformations:

1) Translate the intersection point A(0,c) to the origin, this shifts the line L to L'.
2) Rotate the shifted line L' by –θ degrees so that the line L' aligns with the x-axis.
3) Mirror reflection about x-axis.
4) Rotate the x-axis back by θ degrees
5) Translate the origin back to the intercept point (0,c).

In transformation notation, we have

$$M_L = T_{-v}.R_{-\theta}.M_X.R_{\theta}.T_v \quad , \quad \text{where } v=0I+cJ$$

$$M_L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -c & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & c & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos^2\theta - \sin^2\theta & 2.\cos\theta.\sin\theta & 0 \\ 2.\sin\theta.\cos\theta & \sin^2\theta - \cos^2\theta & 0 \\ -2.c.\sin\theta.\cos\theta & -c.(\sin^2\theta - \cos^2\theta)+c & 1 \end{pmatrix} \quad \text{----------(24)}$$

Let tanθ=m, the standard trigonometry yields $\sin\theta = m/\sqrt{(m^2+1)}$ and $\cos\theta = 1/\sqrt{(m^2+1)}$. Substituting these values for sinθ and cosθ in the equation (24), we have:

$$M_L = \begin{pmatrix} (1-m^2)/(m^2+1) & 2m/(m^2+1) & 0 \\ 2m/(m^2+1) & (m^2-1)/(m^2+1) & 0 \\ -2cm/(m^2+1) & 2c/(m^2+1) & 1 \end{pmatrix} \quad \text{------------(25)}$$

Special cases

1) If we put c = 0 and m=tanθ=0 in the equation (25) then we have the reflection about the line y = 0 i.e. about x-axis. In matrix form:

$$Mx = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{----------(26)}$$

2) If c = 0 and m=tanθ=∞ then we have the reflection about the line x=0 i.e. about y-axis. In matrix form:

$$My = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{---------(27)}$$

4) To get the mirror reflection about the line y = x, we have to put m=1 and c=0.
In matrix form:

$$M_{y=x} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{-----------(28)}$$

5) Similarly, to get the mirror reflection about the line y = – x, we have to put m = –1 and c = 0. In matrix form:

$$M_{y=-x} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{-----------(29)}$$

6) The mirror reflection about the Origin (i.e., an axis perpendicular to the xy plane and passing through the origin).

$$M_{org} = \begin{matrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix} \quad \text{----------(30)}$$

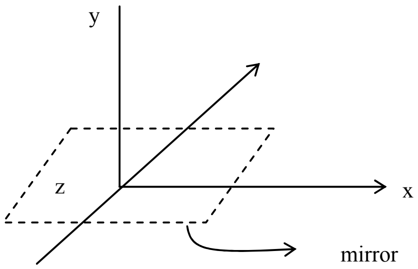


Figure 7(a)

Example 6: Show that two successive reflections about either of the coordinate axes is equivalent to a single rotation about the coordinate origin.

Solution: Let (x, y) be any object point, as shown in *Figure (a).* Two successive reflection of P, either of the coordinate axes, i.e., Reflection about x-axis followed by reflection about y-axis or *vice-versa* can be reprosecuted as:

$$(x, y) \xrightarrow{M_x} (x, -y) \xrightarrow{M_y} (-x, -y) \quad \text{----(i)}$$

$$(x, y) \xrightarrow{M_y} (x, -y) \xrightarrow{M_x} (-x, -y) \quad \text{----(ii)}$$

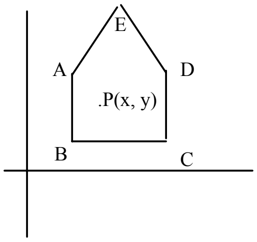
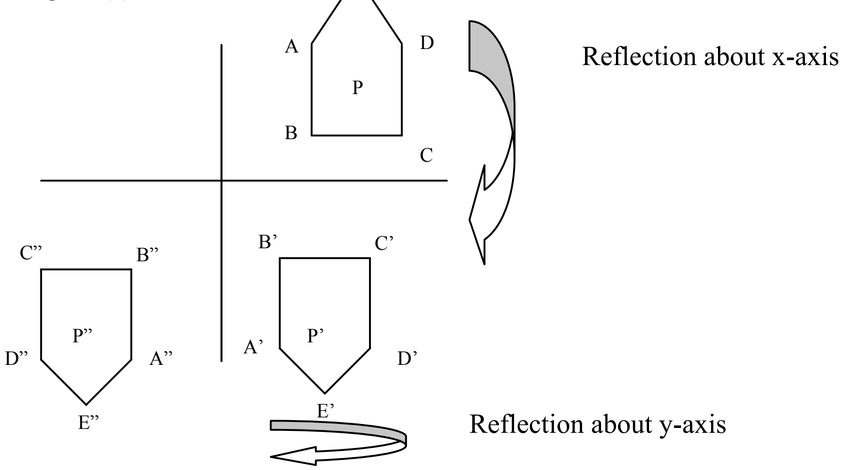


Figure (a)

The effect of (1) and (2) can also be illustrated by the following *Figure (b) and Figure (c)*



Reflection about y-axis

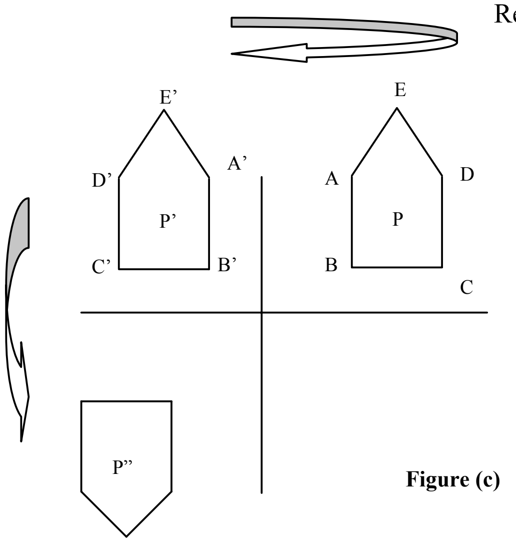


Figure (c)

Reflection about x-axis

From equation (i) and (ii), we can write:

$$(x, y) \longrightarrow (-x, -y) = (x, y)\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \text{(iii)}$$

Equation (3) is the required reflection about the origin. Hence, two successive reflections about either of the coordinate axes is just equivalent to a single rotation about the coordinate origin.

Example 7: Find the transforation matrix for the reflection about the line y = x.

Solution: The transformation for mirror reflection about the line y = x, consists of the following three basic transformations.

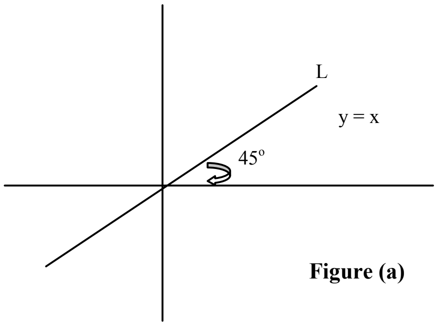


Figure (a)

1) Rotate the line L through 45° in clockwise rotation,
2) Perform the required Reflection about the x-axis.
3) Rotate back the line L by – 45°

i.e.,

$$M_L = R_{45^o} \,.\, M_x \,.\, R_{-45^o}$$

$$= \begin{bmatrix} \text{Cos}\,45^o & -\text{Sin}\,45^o & 0 \\ \text{Sin}\,45^o & \text{Cos}\,45^o & 0 \\ 0 & 0 & 1 \end{bmatrix} \,.\, \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \,.\, \begin{bmatrix} Cos\,45^o & +Sin\,45^o & 0 \\ -Sin\,45^o & Cos\,45^o & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45^o & \sin 45^o & 0 \\ \sin 45^o & -\cos 45^o & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos 45^o & \sin 45^o & 0 \\ -\sin 45^o & \cos 45^o & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 90 & \sin 90 & 0 \\ \sin 90 & -\cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = M_y = x$$

Example 8 : Reflect the diamond-shaped polygon whose vertices are A(–1,0), B(0, –2),C(1,0) and D(0,2) about (a) the horizontal line y=2, (b) the vertical line x=2, and (c) the line y=x+2.

Solution: We can represent the given polygon by the homogeneous coordinate matrix as

$$V=[ABCD] = \begin{pmatrix} -1 & 0 & 1 \\ 0 & -2 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix}$$

a) The horizontal line y=2 has an intercept (0,2) on y axis and makes an angle of 0 degree with the x axis. So m=0 and c=2. Thus, the reflection matrix

$M_L = T_{-v}.R_{-\theta}.M_x.R_{\theta}.T_{-v}$, where $\mathbf{v}=0\mathbf{I}+2\mathbf{J}$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 4 & 1 \end{pmatrix}$$

So the new coordinates A'B'C'D' of the reflected polygon ABCD can be found as:

[A'B'C'D']=[ABCD]. M_L

$$= \begin{pmatrix} -1 & 0 & 1 \\ 0 & -2 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 4 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 4 & 1 \\ 0 & 6 & 1 \\ 1 & 4 & 1 \\ 0 & 2 & 1 \end{pmatrix}$$

Thus, A'=(-1,4), B'=(0,6), C'=(1,4) and D'=(0,2).

b) The vertical line x=2 has no intercept on y-axis and makes an angle of 90 degree with the x-axis. So $m=\tan 90^0=\infty$ and c=0. Thus, the reflection matrix

$M_L = T_{-v}.R_{-\theta}.M_y.R_{\theta}.T_{-v}$, where $\mathbf{v}=2\mathbf{I}$

$$= \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}$$

So the new coordinates A'B'C'D' of the reflected polygon ABCD can be found as:

[A'B'C'D']=[ABCD]. M_L

$$= \begin{pmatrix} -1 & 0 & 1 \\ 0 & -2 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix} . \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 5 & 0 & 1 \\ 4 & -2 & 1 \\ 3 & 0 & 1 \\ 4 & 2 & 1 \end{pmatrix}$$

Thus, A'=(5,0), B'=(4,-2), C'=(3,0) and D'=(4,2)

c) The line y=x+2 has an intercept (0,2) on y-axis and makes an angle of 45^0 with the x-axis. So m=tan45^0=1 and c=2. Thus, the reflection matrix

$$M_L = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ -2 & 2 & 1 \end{pmatrix}$$

The required coordinates A',B', C', and D' can be found as:
[A'B'C'D']=[ABCD]. M_L

$$\begin{pmatrix} -1 & 0 & 1 \\ 0 & -2 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ -2 & 2 & 1 \end{pmatrix}. \quad = \begin{pmatrix} -2 & 1 & 1 \\ -4 & 2 & 1 \\ -2 & 3 & 1 \\ 0 & 2 & 1 \end{pmatrix}$$

Thus, A'=(–2,1), B'=(–4,2), C'=(–2,3) and D'=(0,2)

The effect of the reflected polygon, which is shown in *Figure (a)*, about the line y=2, x=2, and y=x+2 is shown in *Figure (b) - (d)*, respectively.

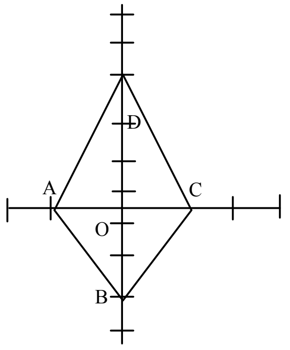


Figure (a)

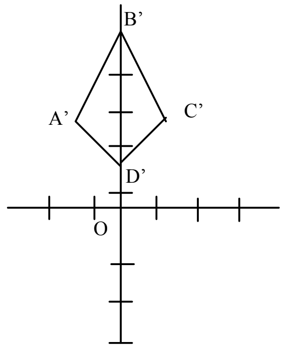


Figure (b)

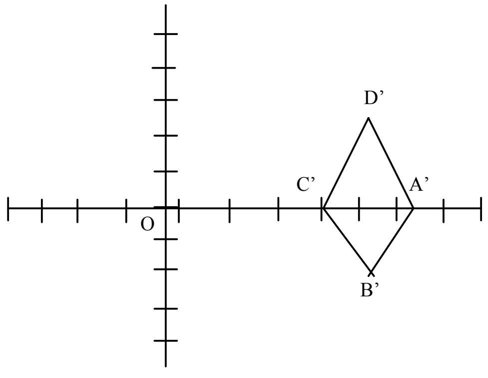


Figure (c)

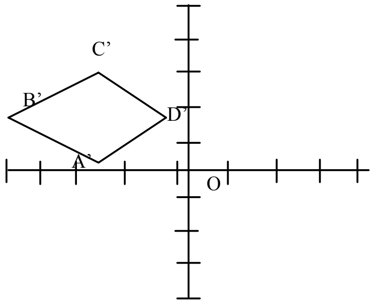


Figure (d)

1.4 HOMOGENEOUS COORDINATE SYSTEMS

Let P(x,y) be any point in 2-D Euclidean (Cartesian) system.

In Homogeneous Coordinate system, we add a third coordinate to a point. Instead of (x,y), each point is represented by a triple (x,y,H) such that H≠0; with the condition that (x1,y1,H1)=(x2,y2,H2) ↔ x1/H1 = x2/H2 ; y1/H1 = y2/H2.

(Here, if we take H=0, then we have point at infinity, i.e., generation of horizons).

Thus, (2,3,6) and (4,6,12) are the same points are represented by different coordinate triples, i.e., each point has many different Homogeneous Coordinate representation.

2-D Euclidian System	**Homogeneous Coordinate System**
Any point (x,y) ⟶	(x,y,1)
	If (x,y,H) be any point in HCS(such that H≠0); Then (x,y,H)=(x/H,y/H,1)
(x/H,y/H) ⟵	(x,y,H)

Now, we are in the position to construct the matrix form for the translation with the use of homogeneous coordinates.

For translation transformation (x,y)→(x+tx,y+ty) in Euclidian system, where tx and ty are the translation factor in x and y direction, respectively. Unfortunately, this way of describing translation does not use a matrix, so it cannot be combined with other transformations by simple matrix multiplication. Such a combination would be desirable; for example, we have seen that rotation about an arbitrary point can be done by a translation, a rotation, and another translation. We would like to be able to combine these three transformations into a single transformation for the sake of efficiency and elegance. One way of doing this is to use homogeneous coordinates. In homogeneous coordinates we use 3x3 matrices instead of 2x2, introducing an additional dummy coordinate H. Instead of (x,y), each point is represented by a triple (x,y,H) such that H≠0; In two dimensions the value of H is usually kept at 1 for simplicity.

Thus, in HCS (x,y,1) → (x+tx,y+ty,1), now, we can express this in matrix form as:

$$(x',y',1)=(x,y,1)\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ t_x & t_y & 1\end{pmatrix}$$

The ***advantage*** of introducing the matrix form of translation is that it simplifies the operations on complex objects, i.e., we can now build complex transformations by multiplying the basic matrix transformations.

In other words, we can say, that a sequence of transformation matrices can be concatenated into a single matrix. This is an effective procedure as it reduces the computation because instead of applying initial coordinate position of an object to each transformation matrix, we can obtain the final transformed position of an object by applying composite matrix to the initial coordinate position of an object. Matrix representation is standard method of implementing transformations in computer graphics.

Thus, from the point of view of matrix multiplication, with the matrix of translation, the other basic transformations such as scaling, rotation, reflection, etc., can also be expressed as 3x3 *homogeneous coordinate matrices*. This can be accomplished by augmenting the 2x2 matrices with a third row (0,0,x) and a third column. That is

$$\begin{bmatrix}0\\0\\1\end{bmatrix} \qquad \begin{pmatrix}a & b & 0\\ c & d & 0\\ 0 & 0 & 1\end{pmatrix}$$

Example 9: Show that the order in which transformations are performed is important by applying the transformation of the triangle ABC by:

(i) Rotating by 45° about the origin and then translating in the direction of the vector (1,0), and

(ii) Translating first in the direction of the vector (1,0), and then rotating by 45° about the origin, where A = (1, 0) B = (0 ,1) and C = (1, 1).

Solution: We can represent the given triangle, as shown in *Figure (a)*, in terms of Homogeneous coordinates as:

$$[ABC] = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

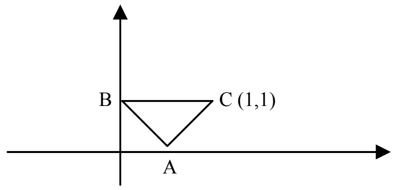


Figure (a)

Suppose the rotation is made in the counter clockwise direction. Then, the transformation matrix for rotation, R_{45^o}, in terms of homogeneous coordinate system is given by:

$$R_{45^o} = \begin{bmatrix} \text{Cos}\,45^o & \text{Sin}\,45^o & 0 \\ -\text{Sin}\,45^o & \text{Cos}\,45^o & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and the Translation matrix, T_v, where V = 1I + 0J is:

$$T_v = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

where t_x and t_y is the translation factors in the x and y directions respectively.

i) Now the rotation followed by translation can be computed as:

$$R_{45^o} \,.\, T_v = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

So the new coordinates $A'B'C'$ of a given triangle ABC can be found as:

$[A'B'C'] = [A\,B\,C] \,.\, R_{45^o} \,.T_v$

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} (1/\sqrt{2}+1) & 1/\sqrt{2} & 1 \\ (-1/\sqrt{2}+1) & 1/\sqrt{2} & 1 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \qquad \text{(I)}$$

implies that the given triangle A(1,0), B (0, 1) C (1, 1) be transformed into $A'\left(\frac{1}{\sqrt{2}}+1, \frac{1}{\sqrt{2}}\right)$, $B'\left(\frac{-1}{\sqrt{2}}+1, \frac{1}{\sqrt{2}}\right)$ and $C'\left(1, \sqrt{2}\right)$, respectively, as shown in *Figure (b)*.

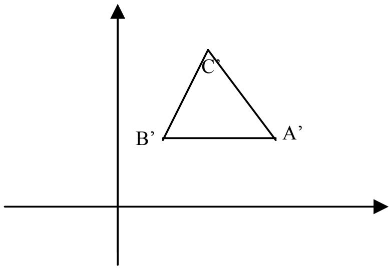


Figure (b)

Similarly, we can obtain the translation followed by rotation transformation as:

$$T_v \,.\, R_{45^o} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1 \end{bmatrix}$$

And hence, the new coordinates $A'B'C'$ can be computed as:

$$[A'B'C'] = [A\,B\,C].T_v \; R_{45^o}$$

$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1 \end{bmatrix} = \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 1 \\ 0 & 2/\sqrt{2} & 1 \\ 1/\sqrt{2} & 3/\sqrt{2} & 1 \end{bmatrix} \qquad \text{(II)}$$

Thus, in this case, the given triangle A(1,0), B(0, 1) and C(1,1) are transformed into $A''\left(2/\sqrt{2}, 2/\sqrt{2}\right)$, $B''\left(0, 2/\sqrt{2}\right)$ *and* $C''\left(\frac{1}{\sqrt{2}}, \frac{3}{\sqrt{2}}\right)$, respectively, as shown in *Figure (c)*.

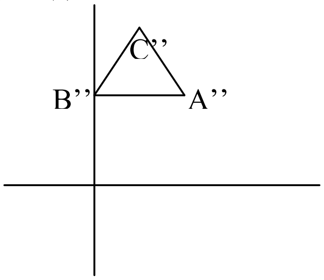


Figure (c)

By (I) and (II), we see that the two transformations do not commute.

☞ Check Your Progress 3

1) Show that transformation matrix (28), for the reflection about the line y=x, is equivalent to the reflection relative to the x-axis followed by a counterclockwise rotation of 90^0.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) Give a single 3x3 homogeneous coordinate transformation matrix, which will have the same effect as each of the following transformation sequences.
   a) Scale the image to be twice as large and then translate it 1 unit to the left.
   b) Scale the x direction to be one-half as large and then rotate counterclockwise by 90^0 about the origin.
   c) Rotate counterclockwise about the origin by 90^0 and then scale the x direction to be one-half as large.
   d) Translate down ½ unit, right ½ unit, and then rotate counterclockwise by 45^0.

3) Obtain the transformation matrix for mirror reflection with respect to the line y=ax, where 'a' is a constant.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

4) Obtain the mirror reflection of the triangle formed by the vertices A(0,3),B(2,0) and C(3,2) about the line passing through the points (1,3) and (–1, –1).

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

1.5 3-D TRANSFORMATIONS

The ability to represent or display a three-dimensional object is fundamental to the understanding of the shape of that object. Furthermore, the ability to rotate, translate, and project views of that object is also, in many cases, fundamental to the understanding of its shape. Manipulation, viewing, and construction of three-dimensional graphic images require the use of three-dimensional *geometric* and *coordinate transformations*. In *geometric transformation*, the coordinate system is fixed, and the desired transformation of the object is done with respect to the coordinate system. In *coordinate* transformation, the object is fixed and the desired transformation of the object is done on the coordinate system itself. These transformations are formed by composing the basic transformations of translation, scaling, and rotation. Each of these transformations can be represented as a matrix transformation. This permits more complex transformations to be built up by use of matrix multiplication or concatenation. We can construct the complex objects/pictures, by instant transformations. In order to represent all these transformations, we need to use homogeneous coordinates.

Hence, if P(x,y,z) be any point in 3-D space, then in HCS, we add a fourth-coordinate to a point. That is instead of (x,y,z), each point can be represented by a Quadruple (x,y,z,H) such that H≠0; with the condition that x1/H1=x2/H2; y1/H1=y2/H2; z1/H1=z2/H2. For two points $(x_1, y_1, z_1, H_1) = (x_2, y_2, z_2, H_2)$ where $H_1 \neq 0$, $H_2 \neq 0$. Thus any point (x,y,z) in Cartesian system can be represented by a four-dimensional vector as (x,y,z,1) in HCS. Similarly, if (x,y,z,H) be any point in HCS then (x/H,y/H,z/H) be the corresponding point in Cartesian system. Thus, a point in three-dimensional space (x,y,z) can be represented by a four-dimensional point as: (x',y',z',1)=(x,y,z,1).[T], where [T] is some transformation matrix and (x',y'z',1) is a new coordinate of a given point (x,y,z,1), after the transformation.

The generalized 4x4 transformation matrix for three-dimensional homogeneous coordinates is:

$$[T]=\begin{pmatrix} a & b & c & w \\ d & e & f & x \\ g & h & I & y \\ l & m & n & z \end{pmatrix} = \left(\begin{array}{c|c} (3x3) & (3x1) \\ \hline (1x3) & (1x1) \end{array}\right) \quad \text{--------(31)}$$

The upper left (3x3) sub matrix produces *scaling, shearing, rotation* and *reflection* transformation. The lower left (1x3) sub matrix produces *translation*, and the upper right (3x1) sub matrix produces a *perspective* transformation, which we will study in the next unit. The final lower right-hand (1x1) sub matrix produces overall *scaling*.

1.5.1 Transformation for 3-D Translation

Let P be the point object with the coordinate (x,y,z). We wish to translate this object point to the new position say, P'(x',y',z') by the translation Vector V=t_x.**I**+t_y.**J**+t_z.**K** , where t_x , t_y and t_z are the translation factor in the x, y, and z directions respectively, as shown in *Figure 8*. That is, a point (x,y,z) is moved to (x+ t_x,y+ t_y,z+ t_z). Thus the new coordinates of a point can be written as:

$$\left.\begin{array}{l} x'=x+t_x \\ y'=y+t_y \\ z'=z+t_z \end{array}\right\} =T_v \quad \text{---------(32)}$$

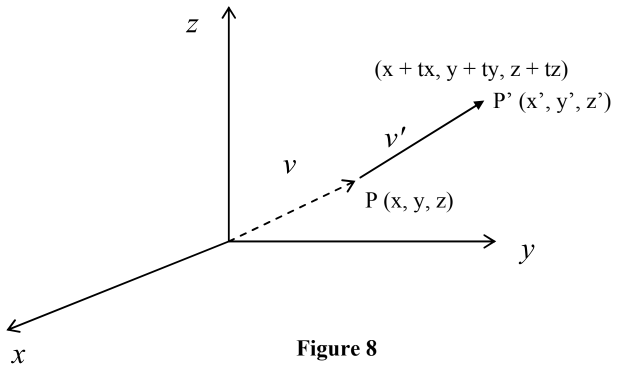


Figure 8

In terms of homogeneous coordinates, equation (32) can be written as

$$(x',y',z',1)=(x,y,z,1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix} \quad \text{--------(33)}$$

i.e., $P'_h = P_h.T_v$ ----------(34)

1.5.2 Transformation for 3-D Rotation

Rotation in three dimensions is considerably more complex then rotation in two dimensions. In 2-D, a rotation is prescribed by an angle of rotation θ and a centre of rotation, say P.

However, in 3-D rotations, we need to mention the angle of rotation and the axis of rotation. Since, we have now three axes, so the rotation can take place about any one of these axes. Thus, we have rotation about x-axis, y-axis, and z-axis respectively.

Rotation about z-axis

Rotation about z-axis is defined by the xy-plane. Let a 3-D point P(x,y,z) be rotated to P'(x',y',z') with angle of rotation θ see *Figure 9*. Since both P and P' lies on xy-plane i.e., z=0 plane their z components remains the same, that is z=z'=0.

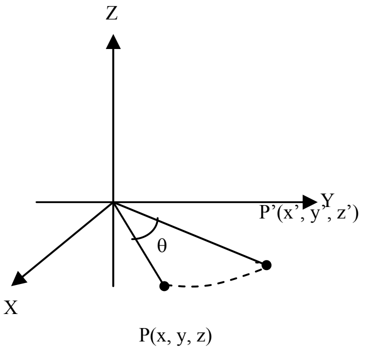


Figure 9

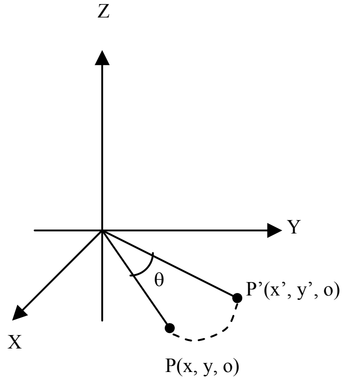


Figure 10

Thus, P'(x'y',0) be the result of rotation of point P(x,y,0) making a positive (anticlockwise) angle ϕ with respect to z=0 plane, as shown in *Figure 10*.

From *figure (10)*,

$P(x,y,0) = P(r.\cos\phi, r.\sin\phi, 0)$
$P'(x',y',0)=P[r.\cos(\phi+\theta), r\sin(\phi+\theta), 0]$

The coordinates of P' are:

$x'=r.\cos(\theta+\phi)=r(\cos\theta\cos\phi - \sin\theta\sin\phi)$
$\quad = x.\cos\theta - y.\sin\theta \quad$ (where $x=r\cos\phi$ and $y=r\sin\phi$)

similarly;

$y'= r\sin(\theta+\phi)=r(\sin\theta\cos\phi + \cos\theta.\sin\phi)$
$\quad = x\sin\theta+y\cos\theta$

Thus,

$$[Rz]_\theta = \left\{\begin{array}{l} x'= x.\cos\theta - y.\sin\theta \\ y'= x\sin\theta + y\cos\theta \\ z'=z \end{array}\right\} \qquad \text{----------(35)}$$

In matrix form,

$$(x'y',z')=(x,y,z)\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{----------(36)}$$

In terms of HCS, equation (36) becomes

$$(x'y',z',1)=(x,y,z,1)\begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{----------(37)}$$

That is, $P'_h = P_h.[Rz]_\theta$ ---------(38)

Rotations about x-axis and y-axis

Rotation about the x-axis can be obtained by cyclic interchange of $x \rightarrow y \rightarrow z \rightarrow x$ in equation (35) of the z-axis rotation i.e.,

$$[Rz]_\theta = \begin{cases} x' = x.\cos\theta - y.\sin\theta \\ y' = x\sin\theta + y\cos\theta \\ z' = z \end{cases}$$

↓ After cyclic interchange of $x \rightarrow y \rightarrow z \rightarrow x$

$$[Rx]_\theta = \begin{cases} y' = y.\cos\theta - z.\sin\theta \\ z' = y.\sin\theta + z.\cos\theta \\ x' = x \end{cases} \qquad \text{--------(39)}$$

So, the corresponding transformation matrix in homogeneous coordinates becomes

$$(x'y',z',1) = (x,y,z,1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P'_h = P_h.[Rx]_\theta$ -------(40)

Similarly, the rotation about y-axis can be obtained by cyclic interchange of $x \rightarrow y \rightarrow z \rightarrow x$ in equation (39) of the x-axis rotation $[Rx]_\theta$ i.e.,

$$[Rx]_\theta = \begin{cases} y' = y.\cos\theta - z.\sin\theta \\ z' = y.\sin\theta + z.\cos\theta \\ x' = x \end{cases}$$

↓ After cyclic interchange of $x \rightarrow y \rightarrow z \rightarrow x$

$$[R_y]_\theta = \begin{cases} z' = z.\cos\theta - x.\sin\theta \\ x' = z.\sin\theta + x.\cos\theta \\ y' = y \end{cases} \qquad \text{-------(41)}$$

So, the corresponding transformation matrix in homogeneous coordinates becomes

$$(x'y',z',1) = (x,y,z,1) \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P' = P.\ [R_y]_\theta$ -------(42)

1.5.3 Transformation for 3-D Scaling

As we have seen earlier, the scaling process is mainly used to change the size of an object. The scale factors determine whether the scaling is a magnification, s>1, or a

reduction, s<1. Two-dimensional scaling, as in equation (8), can be easily extended to scaling in 3-D case by including the z-dimension.

For any point (x,y,z), we move into $(x.s_x,y.s_y,z.s_z)$, where s_x, s_y, and s_z are the scaling factors in the x,y, and z-directions respectively.

Thus, scaling with respect to origin is given by:

$$S_{s_x,s_y,s_z} = \begin{cases} x' = x.s_x \\ y' = y.s_y \\ z' = z.s_z \end{cases} \quad \text{---------(43)}$$

In matrix form,

$$(x'y',z') = (x,y,z) \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \quad \text{---------(44)}$$

In terms of HCS, equation (44) becomes

$$(x'y',z',1) = (x,y,z,1) \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P' = P.\ S_{s_x,s_y,s_z}$ --------(45)

1.5.4 Transformation for 3-D Shearing

Two-dimensional xy- shearing transformation, as defined in equation (19), can also be easily extended to 3-D case. Each coordinate is translated as a function of displacements of the other two coordinates. That is,

$$Sh_{xyz} = \begin{cases} x' = x + a.y + b.z \\ y' = y + c.x + d.z \\ z' = z + e.x + f.y \end{cases} \quad \text{--------(46)}$$

where a,b,c,d,e and f are the shearing factors in the respective directions.

In terms of HCS, equation (46) becomes

$$(x'y',z',1) = (x,y,z,1) \begin{pmatrix} 1 & c & e & 0 \\ a & 1 & f & 0 \\ b & d & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P'_h = P_h.Sh_{xyz}$ -------(47)

Note that the off-diagonal terms in the upper left 3x3 sub matrix of the generalized 4x4 transformation matrix in equation (31) produce shear in three dimensions.

1.5.5 Transformation for 3-D Reflection

For 3-D reflections, we need to know the reference plane, i.e., a plane about which the reflection is to be taken. Note that for each reference plane, the points lying on the plane will remain the same after the reflection.

Mirror reflection about xy-plane

Let P(x,y,z) be the object point, whose mirror reflection is to be obtained about xy-plane(or z=0 plane). For the mirror reflection of P about xy-plane, only there is a change in the sign of z-coordinate, as shown in *Figure (11)*. That is,

$$M_{xy} = \begin{cases} x' = x \\ y' = y \\ z' = -z \end{cases} \quad \text{-------(48)}$$

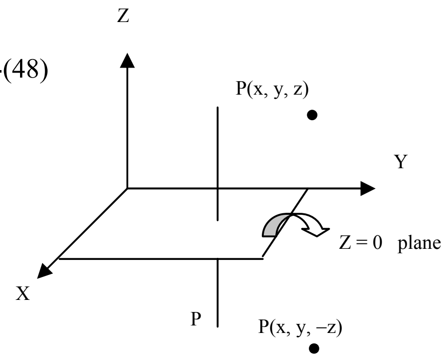


Figure 11

In matrix form,

$$(x'y',z') = (x,y,z) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad \text{----(49)}$$

In terms of HCS (Homogenous coordinate systems), equation (49) becomes

$$(x'y',z',1) = (x,y,z,1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P' = P.M_{xy}$ -----(50)

Similarly, the mirror reflection about yz plane shown in *Figure 12* can be represented as:

$$M_{yz} = \begin{cases} x' = -x \\ y' = y \\ z' = z \end{cases} \quad \text{-----(51)}$$

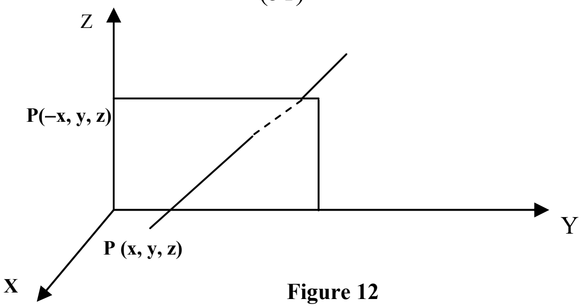


Figure 12

In matrix form,

$$(x'y',z') = (x,y,z) \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{------(52)}$$

In terms of HCS, equation (52) becomes

$$(x'y',z',1) = (x,y,z,1) \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P' = P \cdot M_{yz}$;   ------(53)

and similarly, the reflection about xz plane, shown in *Figure 13*, can be presented as:

$$M_{xz} = \begin{cases} x' = x \\ y' = -y \\ z' = z \end{cases} \qquad \text{-------(54)}$$

In matrix form,

$$(x'\ y', z') = (x, y, z) \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{-------(55)}$$

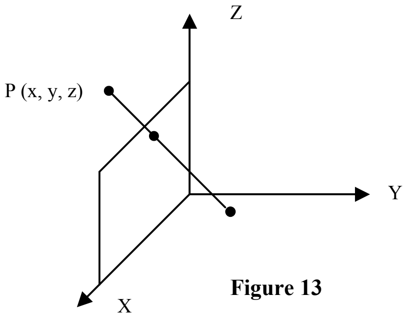


Figure 13

In terms of HCS, equation (55) becomes

$$(x'\ y', z', 1) = (x, y, z, 1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

That is, $P' = P \cdot M_{xz}$ --------(56)

1.6 SUMMARY

In this unit, the following things have been discussed in detail:

- Various geometric transformations such as translation, rotation, reflection, scaling and shearing.
- Translation, Rotation and Reflection transformations are used to manipulate the given object, whereas Scaling and Shearing transformation changes their sizes.
- Translation is the process of changing the position (not the shape/size) of an object w.r.t. the origin of the coordinate axes.
- In 2-D rotation, an object is rotated by an angle θ. There are two cases of 2-D rotation: *case1*- rotation about the origin and *case2*- rotation about an arbitrary point. So, in 2-D, a rotation is prescribed by an angle of rotation θ and a centre of rotation, say P. However, in 3-D rotations, we need to mention the angle of rotation and the axis of rotation.
- Scaling process is mainly used to change the shape/size of an object. The scale factors determine whether the scaling is a magnification, $s>1$, or a reduction, $s<1$.
- Shearing transformation is a special case of translation. The effect of this transformation looks like "pushing" a geometric object in a direction that is parallel to a coordinate plane (3D) or a coordinate axis (2D). How far a direction is pushed is determined by its *shearing factor.*
- Reflection is a transformation which generates the mirror image of an object. For reflection we need to know the reference axis or reference plane depending on whether the object is 2-D or 3-D.
- Composite transformation involves more than one transformation concatenated into a single matrix. This process is also called *concatenation of matrices.* Any transformation made about an arbitrary point makes use of composite transformation such as Rotation about an arbitrary point, reflection about an arbitrary line, etc.
- The use of homogeneous coordinate system to represent the translation transformation in matrix form, extends our N-coordinate system with (N+1) coordinate system.

- The transformations such as translation, rotation, reflection, scaling and shearing can be extended to 3D cases.

1.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Matrix representation are standard method of implementing transformations in computer graphics. But unfortunately, we are not able to represent all the transformations in a (2 x 2) matrix form; such as translation. By using Homogeneous coordinates system (HCS), we can represent all the transformations in matrix form. For translation of point (x, y) $\rightarrow$ (x $+t_x$, y + t_y), it is not possible to represent this transformation in matrix form. But, now in HCS;

$$(x', y', 1) = (x, y, 1)\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

The advantage of introducing the matrix form for translation is that we can now build a complex transformation by multiplying the basic matrix transformation. This is an effective procedure as it reduces the computations.

2) The translation factor, t_x and t_y can be obtained from new old coordinates of vertex C.

$t_x = 6 \; -1 \; = \; 5$

$t_y = 7 \; -1 \; = 6$

The new coordinates [A′ B′ C′ D′] = [A B C D] . T_v

$$\begin{matrix} A' \\ B' \\ C' \\ D' \end{matrix}\begin{bmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ x'_3 & y'_3 & 1 \\ x'_4 & y'_4 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 6 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 6 & 1 \\ 5 & 7 & 1 \\ 6 & 7 & 1 \\ 6 & 6 & 1 \end{bmatrix}$$

Thus A′ = (5, 6), B′ = (5, 7), C′ = (6, 7) and D′ = (6, 6)

3) The new coordinate P′ of a point P, after the Rotation of 45° is:

$P' = P.R_{45^o}$

$$(x', y', 1) = (x, y, 1)\begin{bmatrix} Cos45^o & Sin45^o & 0 \\ -Sin45^o & Cos45^o & 0 \\ 0 & 0 & 1 \end{bmatrix} = (x, y, 1)\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\frac{1}{\sqrt{2}}(x - y), \frac{1}{\sqrt{2}}(x + y, 1)\right] = \left(0, 6/\sqrt{2}, 1\right)$$

Now, this point P′is again translated by $t_x = 5$ and $t_y = 6$. So the final coordinate P″ of a given point P, can be obtained as:

$$(x'', y'', 1) = (x', y', 1) . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 6 & 1 \end{bmatrix}$$

$$= \left(0, 6/\sqrt{2}, 1\right) . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 6 & 1 \end{bmatrix} = \left(5, \frac{6}{\sqrt{6}} + 6, 1\right)$$

Thus P″ (x″, y″) = $(5, \frac{6}{\sqrt{2}} + 6)$

4) $R_\theta = \begin{pmatrix} \text{Cos}\theta & \text{Sin}\theta \\ -\text{Sin}\theta & \text{Cos}\theta \end{pmatrix}$ $R_{-\theta} = \begin{pmatrix} \text{Cos}(-\theta) & \text{Sin}(-\theta) \\ -\text{Sin}(-\theta) & \text{Cos}(\theta) \end{pmatrix} = \begin{pmatrix} \text{Cos}\theta & -\text{Sin}\theta \\ \text{Sin}\theta & \text{Cos}\theta \end{pmatrix}$

$$\because \quad R_\theta . R_{-\theta} = \begin{pmatrix} \text{Cos}\theta & \text{Sin}\theta \\ -\text{Sin}\theta & \text{Cos}\theta \end{pmatrix} \begin{pmatrix} \text{Cos}\theta & -\text{Sin}\theta \\ -\text{Sin}\theta & \text{Cos}\theta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \text{Identity matrix}$$

Therefore, we can say that $R_\theta . R_{-\theta}$ are inverse because $R_\theta . R_{-\theta} = I$. So $R_{-\theta} = R_\theta^{-1}$ i.e., inverse of a rotation by θ degree is a rotation in the opposite direction.

Check Your Progress 2

1) Scaling transformation is mainly used to change the size of an object. The scale factors determines whether the scaling is a compression, S < 1 or a enlargement, S > 1, whereas the effect of shearing is “pushing” a geometric object is a direction parallel to the coordinate axes. Shearing factor determines, how far a direction is pushed.

2) $S_{a,b} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$, $S_{c,d} = \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix}$ and $S_{ac,bd} = \begin{pmatrix} a.c & 0 \\ 0 & b.d \end{pmatrix}$

since

$$S_{a,b}. S_{c,d} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} . \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix} = \begin{pmatrix} a.c & 0 \\ 0 & b.d \end{pmatrix} \quad -(1)$$

$$\text{and } S_{c,d}. S_{a,b} = \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix} . \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} = \begin{pmatrix} c.a & 0 \\ 0 & d.b \end{pmatrix} \quad -(2)$$

from (1) and (2) we can say:

$$S_{a,b}. S_{c,d} = S_{c,d}. S_{a,b} = S_{ac,\, bd}$$

3)

a) Shift an image to the right by 3 units

$$\therefore S = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}$$

b) Shift the image up by 2 units and down by 1 units i.e. $S_x = S_x + 2$ and $S_y = S_y - 1$

$$\therefore\ S = \begin{pmatrix} (S_x+2) & 0 & 0 \\ 0 & (S_y-1) & 0 \\ 0 & 0 & 1 \end{pmatrix}\ \therefore\ S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

c) Move the image down 2/3 units and left 4 units

$$\therefore S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & -2/3 & 1 \end{pmatrix}$$

4) $S_{Sx,\,Sy} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix}$ and $R_{\theta} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$

we have to find out condition under which $S_{Sx,\,Sy}.\ R_\theta = R_{\theta.}\ S_s,\ S_y$

so $S_{Sx,\,Sy}.\ R_\theta = \begin{pmatrix} S_x & o \\ o & S_y \end{pmatrix}.\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} S_x.\cos\theta & S_x.\sin\theta \\ -S_y.\sin\theta & S_y.\cos\theta \end{pmatrix}$ — (1)

and $R_\theta.\ S_{Sx,Sy} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}.\begin{pmatrix} S_x & o \\ o & S_y \end{pmatrix} = \begin{pmatrix} \cos\theta.S_x & Sin\theta.S_y \\ -S_x.Sin\theta & \cos\theta.S_y \end{pmatrix}$ — (2)

In order to satisfy $S_{Sx,\,Sy.}\ R\theta = R\theta.\ S_{Sx,Sy}$

We have $Sy.\ \sin\theta = \sin\theta.S_x \Rightarrow$ either $\sin\theta = 0$ or $\theta = n\pi$, where n is an integer.

$\sin\theta\,(Sy - S_x) = 0$ or $S_x = S_y$ i.e. scaling transform is uniform.

5) No, since $Sh_x(a).\ Sh_y(b) = \begin{pmatrix} 1 & o \\ a & 1 \end{pmatrix}.\begin{pmatrix} 1 & b \\ o & 1 \end{pmatrix} = \begin{pmatrix} 1 & b \\ a & ab+1 \end{pmatrix}$ — (1)

$Sh_y(b).Sh_x(a) = \begin{pmatrix} 1 & b \\ o & 1 \end{pmatrix}\begin{pmatrix} 1 & o \\ a & 1 \end{pmatrix} = \begin{pmatrix} 1+ba & b \\ a & 1 \end{pmatrix}$ — (2)

and $Sh_{xy}(a, b) = \begin{pmatrix} 1 & b \\ a & 1 \end{pmatrix}$

from (1) , (2) and (3), we can say that

$Sh_{xy}(a, b) \neq Sh_x(a)\ .\ Sh_y(b) \neq Sh_y(b).\ Sh_x(a)$

Check Your Progress 3

1) $M_{y=x} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ and

Counter clockwise Rotation of 90°; $R_{90^o} = \begin{bmatrix} \text{Cos}\,90^o & \text{Sin}\,90^o \\ -\text{Sin}\,90^o & \text{Cos}\,90^o \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

We have to show that

$M_{y=x} = M_x \cdot R_{90^o}$

Since $M_x \cdot R_{90^o} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = M_{y = x}$

Hence, a reflection about the line y = x, is equivalent to a reflection relative to the x-axis followed by a counter clockwise rotation of 90°.

2) The required single (3 x 3) homogeneous transformation matrix can be obtained as follows:

a) $T = S_{2,2} \cdot T_{tx-1,\, ty} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix}$

b) $T = S_{s_x + \frac{3}{2}, s_y} \cdot R_{90^o} = \begin{bmatrix} 3/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Cos 90^o & Sin 90^o & 0 \\ -Sin 90^o & Cos 90^o & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$= \begin{bmatrix} 3/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 3/2 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

c) $T = R_{90^o} \cdot S_{S_x + \frac{3}{2}, S_y} = \begin{bmatrix} Cos 90^o & Sin 90^o & 0 \\ -Sin 90^o & Cos 90^o & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

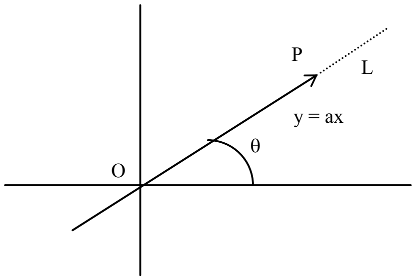
$$= \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 3/2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = T_{tx \frac{-1}{2}, t_y + \frac{1}{2}} \cdot R_{45^o} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{\sqrt{2}} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1\sqrt{2} & 0 & 1 \end{bmatrix}$$

3) Let OP be given line L, which makes an angle θ with respect to



The transformation matrix for reflection about an arbitrary line y = mx + c is (see equation 25).

$$M_L = \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & 0 \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & 0 \\ \frac{-2cm}{m^2+1} & \frac{-2C}{m^2+1} & 1 \end{bmatrix} \quad where\ m = \tan\theta$$

For line y = ax; m = tanθ = a and intercept on y-axis is 0 i.e. c = 0. Thus, transformation matrix for reflection about a line y = ax is:

$$M_L = M_{y=ax} = \begin{bmatrix} \frac{1-a^2}{a^2+1} & \frac{2a}{a^2+1} & 0 \\ \frac{2.a}{a^2+1} & \frac{a^2-1}{a^2+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{where } a = \tan\theta = m$$

4) The equation of the line passing through the points (1,3) and (−1, −1) is obtained as:

y = 2x + 1 (1)

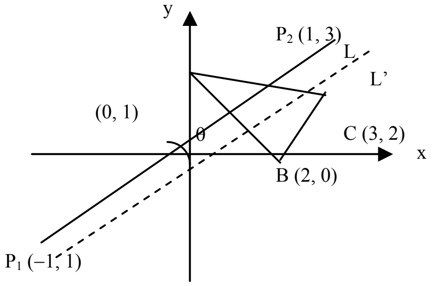


Figure (a)

If θ is the angle made by the line (1) with the positive x-axis, then

$\tan\theta = 2 \Rightarrow \cos\theta = \frac{1}{\sqrt{2}}$ and $\sin\theta\ \frac{2}{\sqrt{5}}$

To obtain the reflection about the line (1), the following sequence of transformations can be performed:

1) Translate the intersection point (0, 1) to the origin, this shift the line L to L′
2) Rotate the shifted line L′ by $-\theta^o$ (i.e. clockwise), so that the L′aligns with the x-axis.

3) Perform the reflection about x-axis.
4) Apply the inverse of the transformation of step (2).
5) Apply the inverse of the transformation of step (1).

By performing step 1 – step 5, we get

$$M_L = T_V \,.\, R_\theta \; M_X \,.\, R_\theta^{-1} \,.\, T_V^{-1}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} . \begin{bmatrix} 1/\sqrt{5} & -2/\sqrt{5} & 0 \\ 2/\sqrt{5} & 1/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} & 0 \\ -2/\sqrt{5} & 1/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -3/\sqrt{5} & 4/5 & 0 \\ 4/5 & 3/5 & 0 \\ -4/5 & 2/5 & 1 \end{bmatrix}$$

So the new coordinates A′B′C′ of the reflected triangle ABC can be found as:
[A′ B′ C′] = [ABC] . M_L

$$= \begin{bmatrix} 0 & 3 & 1 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{bmatrix} . \begin{bmatrix} -3/\sqrt{5} & 4/5 & 0 \\ 4/5 & 3/5 & 0 \\ -4/5 & 2/5 & 1 \end{bmatrix} = \begin{bmatrix} 8/5 & 11/5 & 1 \\ -2 & 2 & 1 \\ -1 & 4 & 1 \end{bmatrix}$$

Thus, $A' = \left(8/5, \dfrac{11}{5}\right)$, B′ = (– 2, 2) and C′ = (–1, 4), which is shown in *Figure (b).*

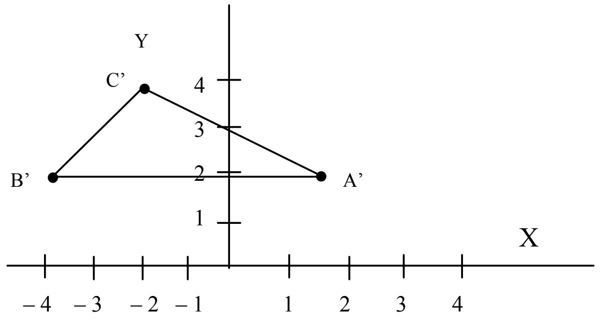


Figure (b)

UNIT 2 VIEWING TRANSFORMATION

Structure **Page Nos.**

2.0	Introduction	44
2.1	Objectives	45
2.2	Projections	45
	2.2.1 Parallel Projection	47
	2.2.1.1 Orthographic and Oblique Projections	51
	2.2.1.2 Isometric Projections	61
	2.2.2 Perspective Projections	65
2.3	Summary	83
2.4	Solutions/Answers	84

2.0 INTRODUCTION

In unit 1, we have discussed the geometric transformations such as Translation, Rotation, Reflection, Scaling and Shearing. Translation, Rotation and Reflection transformations are used to manipulate the given object, whereas Scaling and Shearing transformations are used to modify the shape of an object, either in 2-D or in 3-Dimensional.

A transformation which maps 3-D objects onto 2-D screen, we are going to call it *Projections.* We have two types of Projections namely, *Perspective projection* and *Parallel projection*. This categorisation is based on the fact whether rays coming from the object converge at the centre of projection or not. If, the rays coming from the object converge at the centre of projection, then this projection is known as *Perspective projection*, otherwise it is *Parallel projection*. In the case of parallel projection the rays from an object converge at infinity, unlike perspective projection where the rays from an object converge at a finite distance (called COP).

Parallel projection is further categorised into *Orthographic* and *Oblique projection*. Parallel projection can be categorized according to the angle that the direction of projection makes with the projection plane If the direction of projection of rays is perpendicular to the projection plane then this parallel projection is known as *Orthographic projection* and if the direction of projection of rays is not perpendicular to the projection plane then this parallel projection is known as *Oblique projection*. The orthographic (perpendicular) projection shows only the front face of the given object, which includes only two dimensions: length and width. The oblique projection, on the other hand, shows the front surface and the top surface, which includes three dimensions: length, width, and height. Therefore, an oblique projection is one way to show all three dimensions of an object in a single view.

Isometric projection is the most frequently used type of *axonometric projection*, which is a method used to show an object in all three dimensions (length, width, and height) in a single view. Axonometric projection is a form of orthographic projection in which the projectors are always perpendicular to the plane of projection.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- define the projection;
- categorize various types of Perspective and Parallel projections;
- develop the general transformation matrix for parallel projection;
- describe and develop the transformation for Orthographic and oblique parallel projections;
- develop the transformations for multiview (front, right, top, rear, left and bottom view) projections;
- define the foreshortening factor and categorize the oblique projection on the basis of foreshortening factors;
- derive the transformations for general perspective projection;
- describe and derive the projection matrix for single-point, two-point and three-point perspective transformations, and
- identify the vanishing points.

2.2 PROJECTIONS

.

Given a 3-D object in a space, Projection can be defined as a mapping of 3-D object onto 2-D viewing screen. Here, 2-D screen is known as Plane of projection or view plane, which constitutes the display surface. The mapping is determined by projection rays called the projectors. Geometric projections of objects are formed by the intersection of lines (called projectors) with a plane called plane of projection /view plane. Projectors are lines from an arbitrary point, called the centre of projection (COP), through each point in an object. *Figure 1* shows a mapping of point P(x,y,z) onto its image P′(x’,y’,z’) in the view plane.

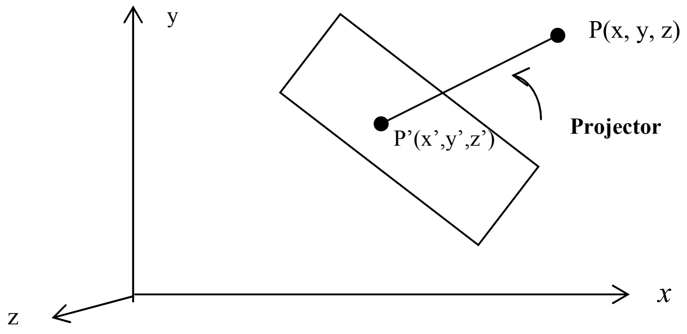


Figure 1

If, the COP (Center of projection) is located at finite point in the three-space, the result is a perspective projection. If the COP is located at infinity, all the projectors are parallel and the result is a parallel projection. *Figure 2(a)-(b)* shows the difference between parallel and perspective projections. In *Figure 2(a)*, **ABCD is projected to A’B’C’D’ on the plane of projection and O is a COP**. In the case of parallel projection the rays from an object converges at infinity, the rays from the object become parallel and will have a direction called ”direction of projection”.

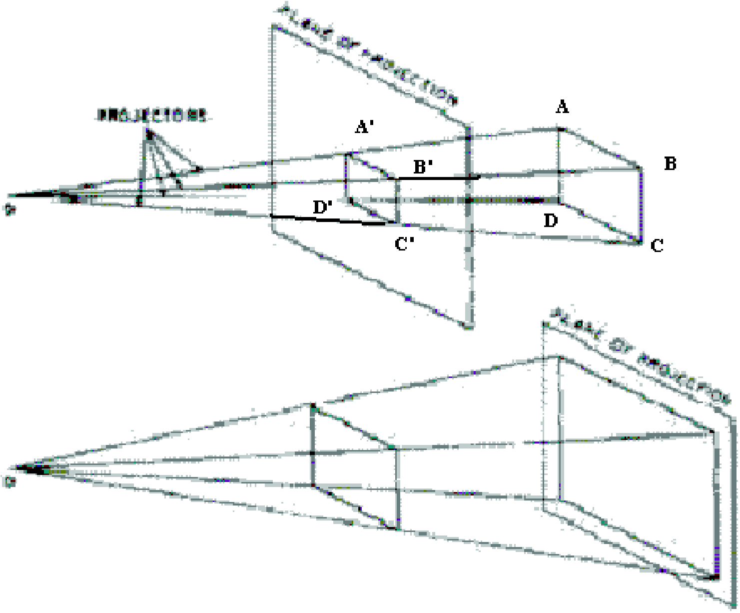


Figure 2(a): Perspective projection

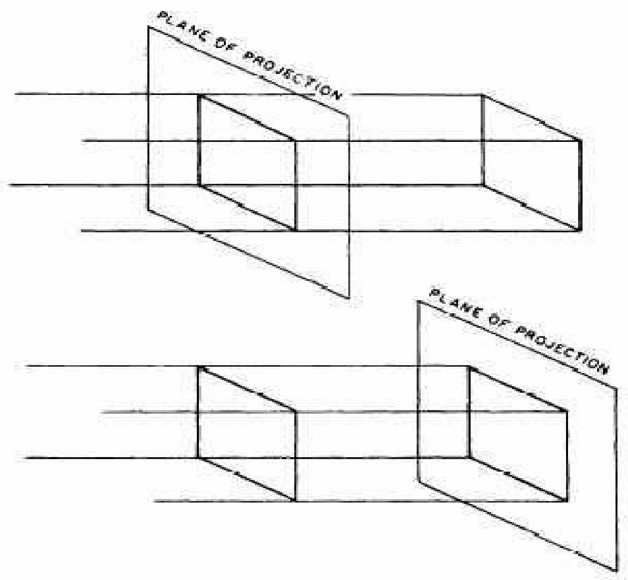


Figure 2(b): Parallel projection

Taxonomy of Projection

There are various types of projections according to the view that is desired. The following *Figure 3* shows taxonomy of the families of *Perspective* and *Parallel* Projections. This categorisation is based on whether the rays from the object converge at COP or not and whether the rays intersect the projection plane perpendicularly or not. The former condition separates the perspective projection from the parallel projection and the latter condition separates the Orthographic projection from the Oblique projection.

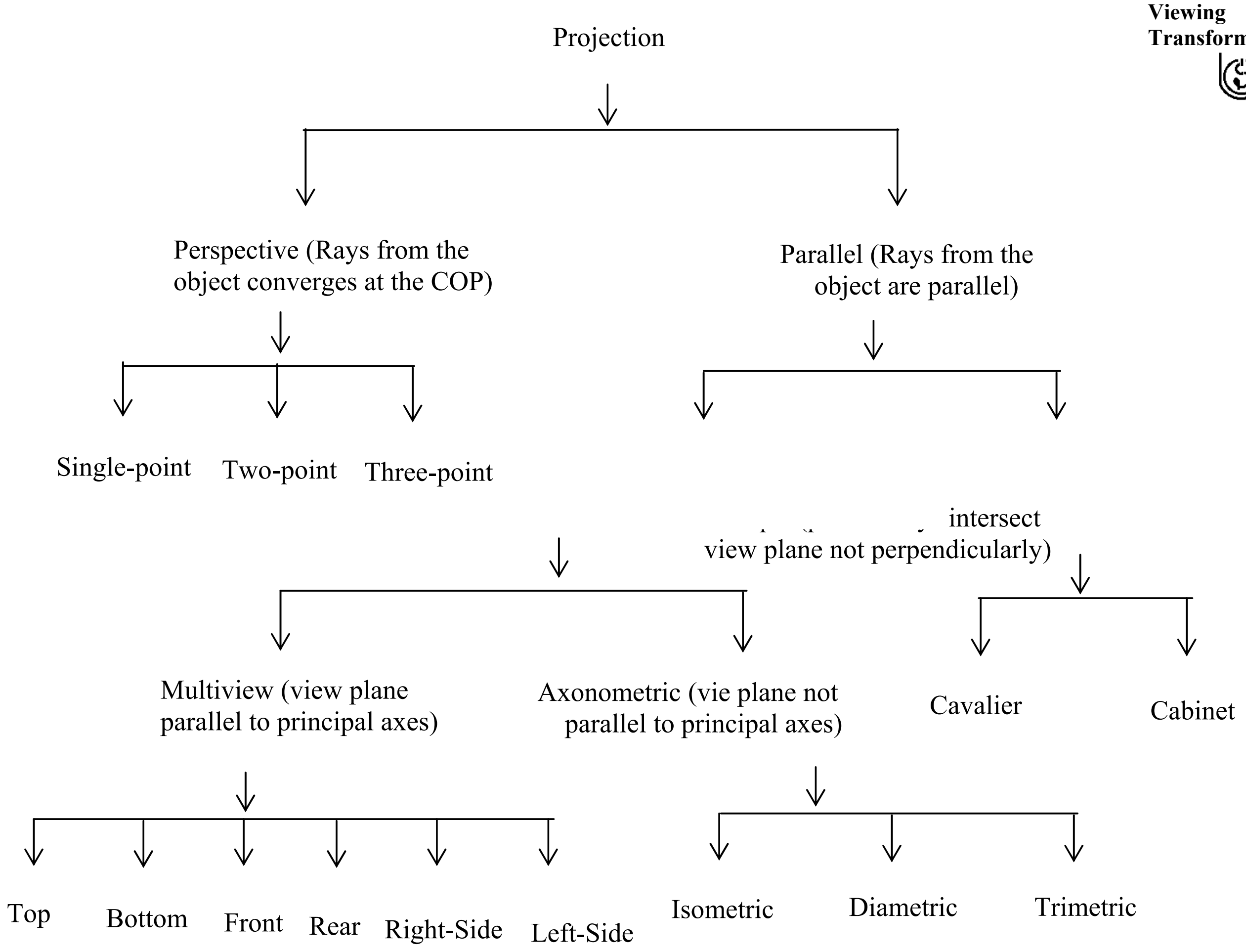


Figure 3: Taxonomy of projection

The direction of rays is very important only in the case of Parallel projection. On the other hand, for Perspective projection, the rays converging at the COP, they do not have a fixed direction i.e., each ray intersects the projection plane with a different angle. For Perspective projection the direction of viewing is important as this only determines the occurrence of a vanishing point.

2.2.1 Parallel Projection

Parallel projection methods are used by engineers to create working drawings of an object which preserves its true shape. In the case of parallel projection the rays from an object converge at infinity, unlike the perspective projection where the rays from an object converse at a finite distance (called COP).

If the distance of COP from the projection plane is infinite then parallel projection (all rays parallel) occurs i.e., when the distance of COP from the projection plane is infinity, then all rays from the object become parallel and will have a direction called "**direction of projection**". It is denoted by **d**=(d1,d2,d3), which means **d** makes unequal/equal angle with the positive side of the x,y,z axes.

Parallel projection can be categorised according to the angle that the direction of projection makes with the projection plane. For example, in Isometric projection, the direction of projection **d**=(d1,d2,d3) makes equal angle (say α) with all the three-principal axes (see *Figure 4*).

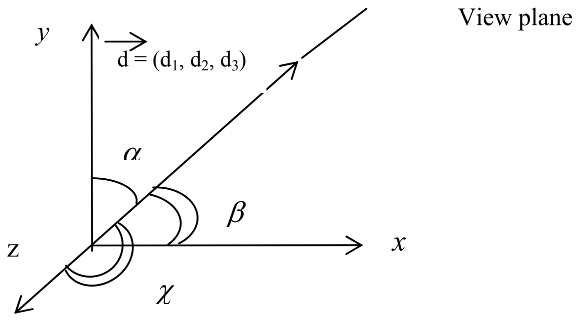


Figure 4: Direction of projection

Rays from the object intersect the plane before passing through COP. In parallel projection, image points are found as the intersection of view plane with a projector (rays) drawn from the object point and having a fixed direction.(see *Figure 5*).

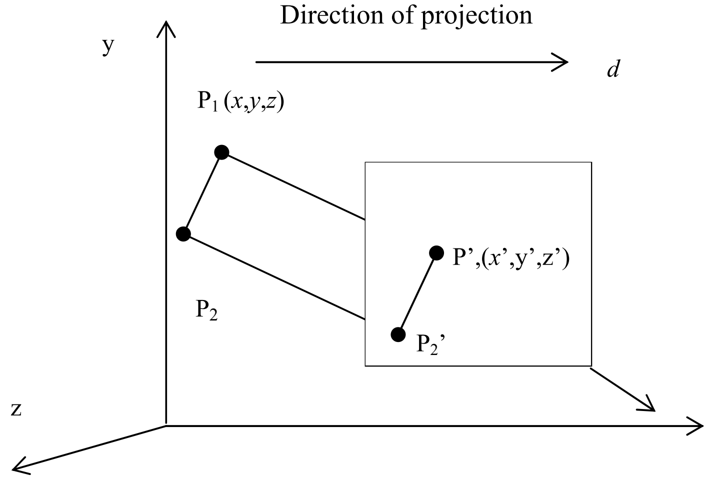


Figure 5: Parallel projection

Parallel rays from the object may be perpendicular or may not be perpendicular to the projection plane. If the direction of projection **d**=(d1,d2,d3) of the rays is perpendicular to the projection plane (or **d** has the same direction as the view plane normal **N**), we have *Orthographic projection* otherwise *Oblique projection.*

Orthographic projection is further divided into *Multiview projection* and *axonometric projection*, depending on whether the direction of projection of rays is parallel to any of the principal axes or not. If the direction of projection is parallel to any of the principal axes then this produces the *front*, *top* and *side views* of a given object, also referred to as *multiview* drawing (see *Figure 8*).

Axonometric projections are orthographic projection in which the direction of projection is not parallel to any of the 3 principle axes. *Oblique* projections are non-orthographic parallel projections i.e., if the direction of projection **d**=(d1,d2,d3) is not perpendicular to the projection plane then the parallel projection is called an *Oblique* projection.

Transformation for parallel projection

Parallel projections (also known as Orthographic projection), are projections onto one of the coordinate planes x = 0, y = 0 or z = 0. The standard transformation for parallel (orthographic) projection onto the xy-plane (i.e. z=0 plane) is:

$P_{par,z}$ =

$$\begin{cases} x' = x \\ y' = y \\ z' = 0 \end{cases}$$

In matrix form:

$$P_{par,z} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{------------------------------(1)}$$

Thus, if P(x,y,z) be any object point in space, then projected point P'(x'y'z') can be obtained as:

$$(x', y', z, 1) = (x, y, z, 1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{------------------------------(2)}$$

$$P'_h = P_h . P_{par,z} \quad \text{------------------------------(3)}$$

Example1: Derive the general transformation of parallel projection onto the xy-plane in the direction of projection d=aI+bJ+cK.

Solution: The general transformation of parallel projection onto the xy-plane in the direction of projection d=a**I**+b**J**+c**K,** is derived as follows(see *Figure a*):

Let P(x,y,z) be an object point, projected to P'(x',y',z') onto the z'=0 plane. From *Figure (a)* we see that the vectors **d** and **PP'** have the same direction. This means that

PP'=k.**d** , comparing components, we have:

x'-x=k.a
y'-y=k.b
z'-z=k.c

Since z'=0 on the projection plane, we get k=-z/c.
Thus,

x'=x-a.z/c
y'=y-b.z/c
z'=0

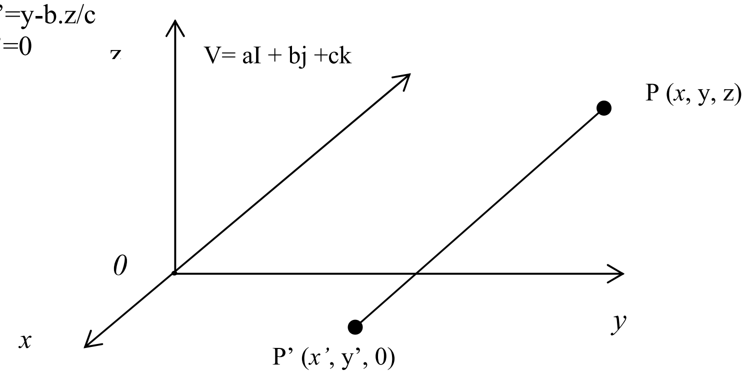


Figure (a)

In terms of homogeneous coordinates, this equation can be written as:

$$(x',y',z'1) = (x, y, z, 1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -a/c & -b/c & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{-----------------------------(4)}$$

That is, $P'_h = P_h.P_{par,z}$, where $P_{par,z}$ is the parallel projection with the direction of projection **d** along the unit vector k.

Example 2: Derive the general transformation for parallel projection onto a given view plane, where the direction of projection d=aI+bJ+cK is along the normal $N=n_1I+n_2J+n_3K$ with the reference point $R_0(x_0,y_0,z_0)$.

Solution: The general transformation for parallel projection onto the xy-plane in the direction of projection *Figure (b)*
v = a I + bJ + ck, denoted by P par, V, N, Ro, consists of the following steps:

1) Translate the view reference point R_o of the view plane to the origin, by $T\text{-}R_o$
2) Perform an alignment transformation An so that the view normal vector N of the view points in the direction K of the normal to the xy-plane. The direction of projection vector V is transformed to new vector V' = AnV.
3) Project onto the xy-plane using P par, v'
4) Align k back to N, using An.
5) Translate the origin back to Ro, by T_{Ro}

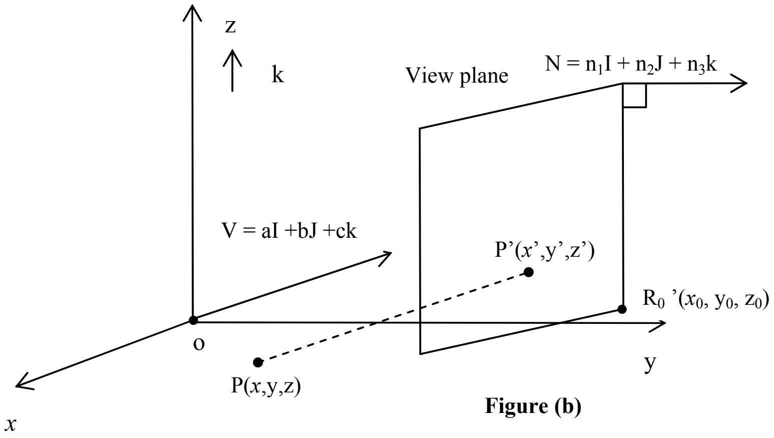


Figure (b)

So

$P_{par}, V, N, R_o = T\text{–}Ro\ A_N^{-1} . P_{par, v}'. An. T_{Ro}$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\lambda}{|N|} & \frac{-n_1n_2}{|N|} & \frac{-n_1n_3}{} & 0 \\ 0 & \frac{n_3}{\lambda} & \frac{n_2}{\lambda} & 0 \\ \frac{n_1}{|N|} & \frac{n_2}{|N|} & \frac{n_3}{|N|} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{-a}{c} & 1 & 0 & 0 \\ \frac{-a}{c} & \frac{-b}{c} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\lambda}{|N|} & 0 & \frac{n_2}{|N|} & 0 \\ \frac{-n_1 n_2}{|N|} & \frac{n_3}{\lambda} & \frac{n_2}{|N|} & 0 \\ \frac{-n_1 n_3}{\lambda |N|} & \frac{n_2}{|N|} & \frac{n_3}{|N|} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & z_0 & 1 \end{pmatrix}$$

where λ =

$\sqrt{n_2^2 + n_3^2}$ and $\lambda \neq 0$.

After multiplying all the matrices, we have:

$$P\ par, V, N, R_0 = \begin{pmatrix} d_1 - an_1 & -bn_1 & -cn_1 & 0 \\ -an_2 & d_1 - bn_2 & -cn_2 & 0 \\ -an_3 & -bn_3 & d_1 - cn_3 & 0 \\ ad_0 & bd_0 & c.do & d_1 \end{pmatrix} \text{---------------------------(5)}$$

Where $d_0 = n_1 x_0 + n_2 y_0 + n_3 z_0$ and
$d_1 = n_1 a + n_2 b + n_3 c$

Note: Alignment transformation, An, refer any book for computer graphic.

2.2.1.1 Orthographic and Oblique Projections

Orthographic projection is the simplest form of parallel projection, which is commonly used for engineering drawings. They actually show the 'true' size and shape of a single plane face of a given object.

If the direction of projection **d**=(d1,d2,d3)has the direction of view plane normal **N** (or **d** is perpendicular to view plane)**,** the projection is said to be *orthographic*. Otherwise it is called *Oblique* projection. The *Figure 6* shows the orthographic and oblique projection.

We can see that the orthographic (perpendicular) projection shows only front surface of an object, which includes only two dimensions: length and width. The oblique projection, on the other hand, shows the front surface and the top surface, which includes three dimensions: length, width, and height. Therefore, an oblique projection is one way to show all three dimensions of an object in a single view

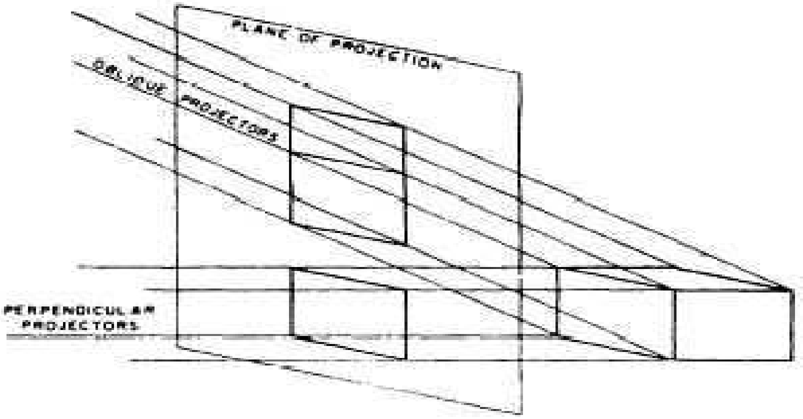


Figure 6: Orthographic and oblique projection

Orthographic projections are projections onto one of the coordinate planes x=0, y=0or z=0. The matrix for orthographic projection onto the z=0 plane (i.e. xy-plane) is:

$$P_{par,z} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{------------------------------}(6)$$

Note that the z-column (third column) in this matrix is all zeros. That is for orthographic projection onto the z=0 plane, the z-coordinates of a position vector is set to zero. Similarly, we can also obtain the matrices for orthographic projection onto the x=0 and y=0 planes as:

$$P_{par,x} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{------------------------------}(7)$$

and

$$P_{par,y} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{------------------------------}(8)$$

For example, consider the object given in *Figure 6(a).* The orthographic projections of this object onto the x=0, y=0 and z=0 planes from COP at infinity on the +x-, +y- and +z-axes are shown in *Figure 7* (b)-(d).

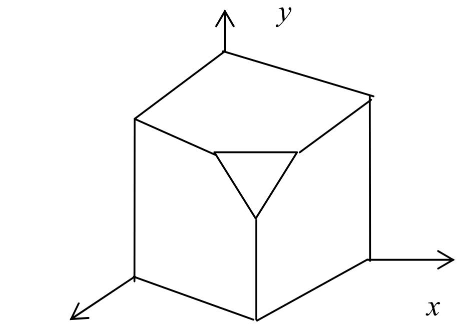


Figure 7(a)

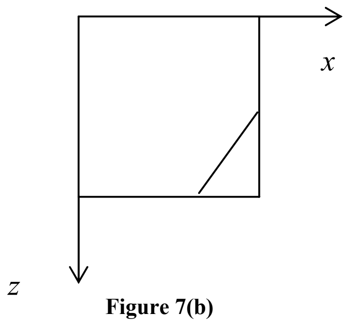


Figure 7(b)

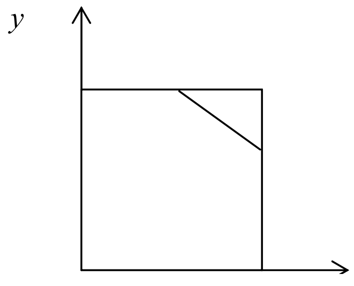


Figure 7(c)

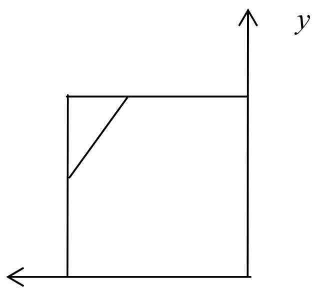


Figure 7(d)

A single orthographic projection does not provide sufficient information to visually and practically reconstruct the shape of an object. Thus multiple orthographic projections are needed (known as *multiview drawing*). In all, we have 6 views:

1) Front view
2) Right-side view
3) Top-view
4) Rear view
5) Left-side view
6) Bottom view

The *Figure 8* shows all 6 views of a given object.

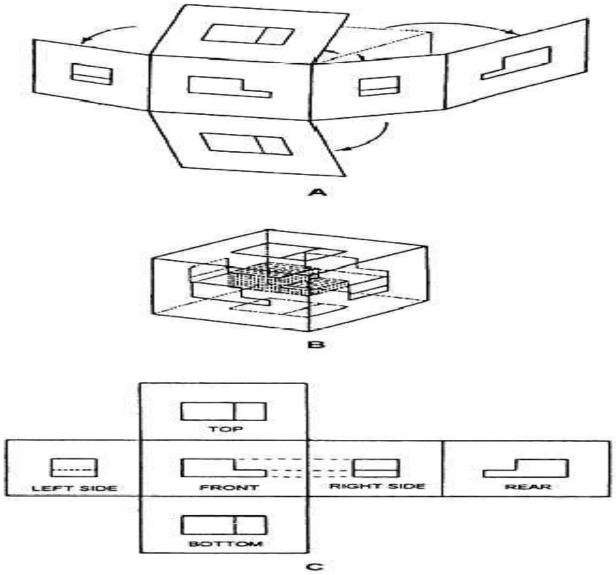


Figure 8: Multiview orthographic projection

The front, right-side and top views are obtained by projection onto the z=0, x=0 and y=0 planes from COP at infinity on the +z-, +x-, and +y-axes.

The rear, left-side and bottom view projections are obtained by projection onto the z=0, x=0, y=0 planes from COP at infinity on the –z-, -x and –y-axes(see *Figure 8*). All six views are normally not required to convey the shape of an object. The front, top and right-side views are most frequently used.

The direction of projection of rays is shown by arrows in *Figure 9*.

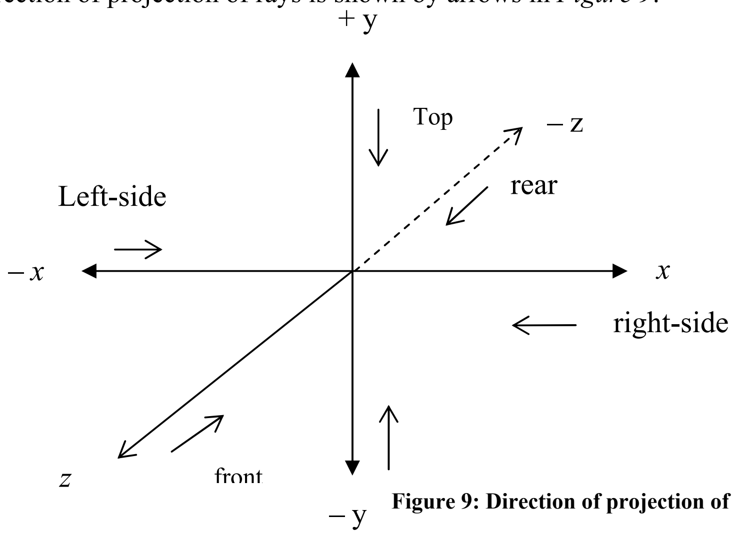


Figure 9: Direction of projection of rays in multiview drawing

The projection matrices for the front, the right-side and top views are given by:

$P_{front} = P_{par,z}$=diag(1,1,0,1)

$P_{right} = P_{par,x}$=diag(0,1,1,1)

$P_{top} = P_{par,y}$=diag(1,0,1,1)

It is important to note that the other remaining views can be obtained by combinations of reflection, rotation and translation followed by projection onto the z=0 plane from the COP at infinity on the +z-axis. For example: the rear view is obtained by reflection through the z=0 plane, followed by projection onto the z=0 plane.

$P_{rear}=M_{xy}.\ P_{par,z}$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{------------------------(9)}$$

Similarly, the left-side view is obtained by rotation about the y-axis by $+90^0$, followed by projection onto the z=0 plane.

$P_{left}=[R_y]_{90^0}.P_{par,z}$

$$= \begin{pmatrix} \cos 90 & 0 & -\sin 90 & 0 \\ 0 & 1 & 0 & 0 \\ \sin 90 & 0 & \cos 90 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{------------------(10)}$$

And the bottom view is obtained by rotation about the x-axis by -90^0, followed by projection onto the z=0 plane.

$P_{bottom}=[R_x]_{90^0}.P_{par,z}$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90) & \sin(-90) & 0 \\ 0 & -\sin(-90) & \cos(-90) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{-----------(11)}$$

Example 3: Show all the six views of a given object shown in following *Figure*. The vertices of the object are A(4,0,0), B(4,4,0), C(4,4,8), D(4, 0, 4), E (0,0,0), F(0,4,0), G(0,4,8), H(0,0,4).

Solution: We can represent the given object in terms of Homogeneous-coordinates of its vertices as:

$$V= [ABCDEFGH] = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix} \begin{pmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{pmatrix}$$

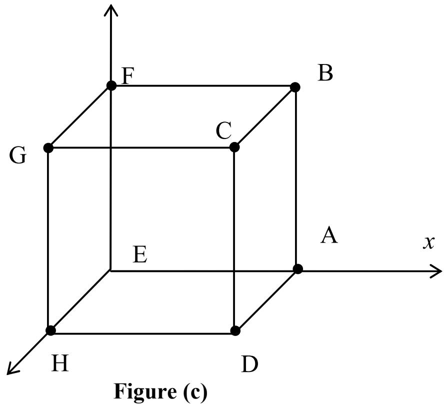


Figure (c)

(1) If we are viewing from the front, then the new coordinate of a given object can be found as:

$P'_{n},z = P_n . P_{front}$

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{pmatrix} x'1 & y'1 & 1 \\ x'2 & y'2 & 1 \\ x'3 & y'3 & 1 \\ x'4 & y'4 & 1 \\ x'5 & y'5 & 1 \\ x'6 & y'6 & 1 \\ x'7 & y'7 & 1 \\ x'8 & y'8 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{pmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & \end{pmatrix}$$

from matrix, we can see that
A′ = D′, B′ = C′, E′ = H′, F′ = G′, Thus we can see only C′D′G′H′
as shown in *Figure d*

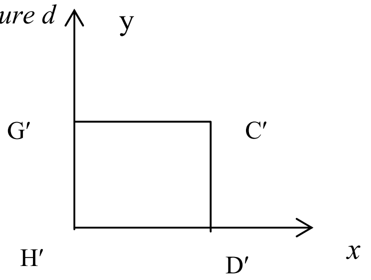


Figure d

(2) If we are viewing from right-side, then

$$P'_{n,x} = V._{\text{Pright}} = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix}\begin{pmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 4 & 1 \end{pmatrix} . \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{pmatrix}$$

Here, we can see that A’ =E’, B’ = F’, C’ = G’ and D’ = H’.
Thus, we can see only A’B’C’D’ as shown in *Figure e.*

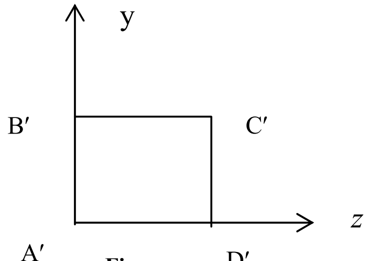


Figure e

(3) if we are viewing from top, then

$$P'_{n,y} = P_n . P_{top} = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix}\begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 4 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 4 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{bmatrix} 4 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 4 & 0 & 8 & 1 \\ 4 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 8 & 1 \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

Here, we can see that A' = B', E' = F', C' ≠ D' and G' ≠ H'
Thus we can see only the square B'F'G'C' but the line H'D' is hidden an shown in *Figure f.*

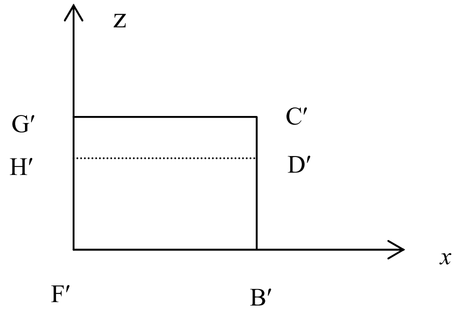


Figure f

Similarly we can also find out the other side views like, rear left-side and bottom using equation – 1, 2, 3

☞ Check Your Progress 1

1) Define the following terms related with Projections with a suitable diagram:

a) Center of Projection (COP)
b) Plane of projection/ view plane
c) Projector
d) Direction of projection

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

2) Categories the various types of parallel and perspective projection.

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

3) In orthographic projection
   a) Rays intersect the projection plane.
   b) The parallel rays intersect the view plane not perpendicularly.
   c) The parallel rays intersect the view plane perpendicularly.
   d) none of these

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

Oblique projection

If the direction of projection **d**=(d1,d2,d3) of the rays is not perpendicular to the view plane(or **d** does not have the same direction as the view plane normal **N**), then the parallel projection is called an *Oblique projection* (see *Figure 10*).

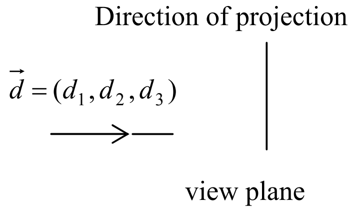


Figure 10 (a): Oblique projection

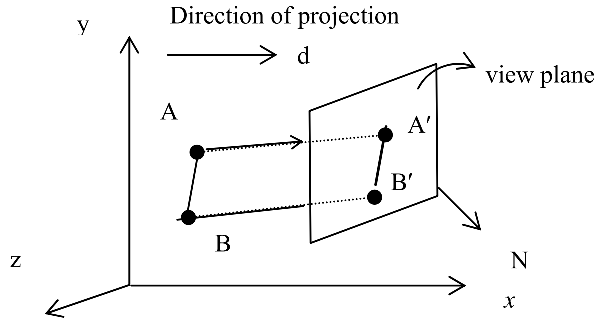


Figure 10 (b): Oblique projection

In oblique projection only the faces of the object parallel to the view plane are shown at their true size and shape, angles and lengths are preserved for these faces only. Faces not parallel to the view plane are discarded.

In Oblique projection the line perpendicular to the projection plane are *foreshortened* (shorter in length of actual lines) by the direction of projection of rays. The direction of projection of rays determines the amount of foreshortening. The change in length of the projected line (due to the direction of projection of rays) is measured in terms of foreshortening factor, f.

Foreshortening factors w.r.t. a given direction

Let AB and CD are two given line segments and direction of projection **d**=(d1,d2,d3). Also assumed that AB $\|$ CD $\|$ **d** . Under parallel projection, let AB and CD be projected to A'B' and C'D', respectively.

Observation:

i) A'B' $\|$ C'D' will be true, i.e. Parallel lines are projected to parallel lines, under parallel projection.

ii) |A'B'|/|AB| = |C'D'|/|CD| must be true, under parallel projection.

This ratio (projected length of a line to its true length) is called the foreshortening factor w.r.t. a given direction.

Mathematical description of an Oblique projection (onto xy-plane)

In order to develop the transformation for the oblique projection, consider the *Figure 10*. This *figure* shows an oblique projection of the point A (0, 0, 1) to position A'(x',y',0) on the view plane (z=0 plane). The direction of projection **d**=(d1,d2,d3).

Oblique projections (to xy-plane) can be specified by a number f and an angle θ. The number f, known as foreshortening factor, indicates the ratio of projected length OA'of a line to its true length. Any line L perpendicular to the xy-plane will be foreshortened after projection.

θ is the angle which the projected line OA'(of a given line L perpendicular to xy-plane) makes with the positive x-axis.

The line OA is projected to OA'. The length of the projected line from the origin =|OA'|

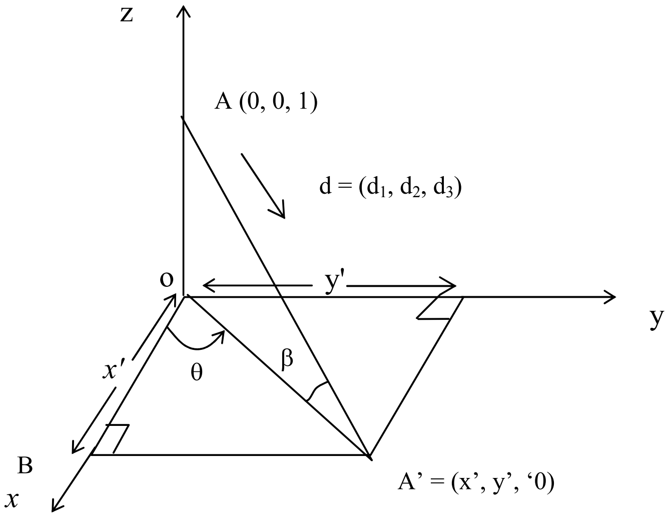


Figure 11: Oblique projection

Thus, foreshortening factor, f=|OA'|/|OA|=|OA'|, in the z-direction
From the triangle OAP', we have,

OB=x'=f.cosθ
BA'=y'=f.sinθ

When f = 1, then obligue projection is known as Cavalier projection

Given θ = 45°, then we have

$$P_{cav} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

When f = ½ then obligue projection is called a cabinet projection.

Here θ = 30° (Given), we have

$$\text{Pcab} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \sqrt{3}/4 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

we can represent a given unit cube in terms of Homogeneous coordinates of the

$$\text{vertices as:}V = [A\ B\ C\ D\ E\ F\ G\ H] = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

i) To draw the cavalier projection, we find the image coordinates of a given unit cube as follows:

$$P' = V.\ P_{cav} = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ \sqrt{2}/2 & (1+\frac{\sqrt{2}}{2} & 0 & 1 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 & 1 \\ (1+\sqrt{2}/2 & \sqrt{2}/2 & 0 & 1 \\ (1+\sqrt{2}/2 & (1+\sqrt{2}/2 & 0 & 1 \end{bmatrix}$$

Hence, the image coordinate are:

A' = (0, 0, 0), B' = (1, 0, 0), C' = (1, 1, 0), D' = (0, 1, 0) E' = (√2/2 , 1 + √2/2, 0)
F' = (√2/2, √2/2, 0), G' = (1 + √2/2, √2/2, 0), H' = (1 + √2/2, 1 + √2/2, 0)

Thus, cavalier projection of a unit cube is shown in *Figure 11(a)*.

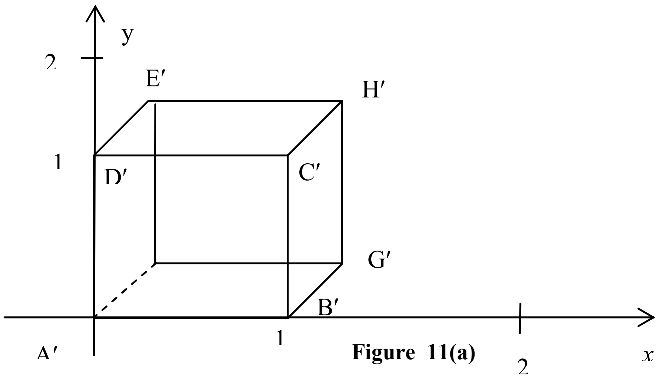


Figure 11(a)

To determine projection matrix for oblique projection, we need to find the direction vector **d.** Since vector **PP'** and vector **d** have the same direction. Thus, **PP'=d**

Thus, $x' - 0 = d_1 = f.\cos\theta$
$y' - 0 = d_2 = f.\sin\theta$
$z' - 1 = d_3$

As z'=0 on the xy-plane, $d_3 = -1$,

Since, Oblique projection is a special case of parallel projection, thus, we can transform the general transformation of parallel projection for Oblique projection as follows:

$$P_{Oblique} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -d_1/d_3 & -d_2/d_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ f.\cos\theta & f.\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{-----------------(12)}$$

Where, f=foreshortening factor, i.e., the projected length of the z-axis unit vector. If β is the angle

Between the Oblique projectors and the plane of projection then,
$1/f = \tan(\beta)$, i.e., $f = \cot(\beta)$ ------------ (13)

θ=angle between the projected line with the positive x-axis.

Special cases:

1) If f=0, then cot (β)=0 that is $\beta=90^0$, then we have an Orthographic projection.
2) If f=1, the edge perpendicular to projection plane are not foreshortened, then $\beta=\cot^{-1}(1)=45^0$ and this Oblique projection is called *Cavalier* projection.
3) If f=1/2 (the foreshortening is half of unit vector), then $\beta=\cot^{-1}(1/2)=63.435^0$ and this Oblique projection is called *Cabinet* projection.

Note: The common values of θ are **30° and 45°. the values of (180° –** θ) is also acceptable.

The *Figure 12* shows an Oblique projections for foreshortening factor f=1,7/8,3/4,5/8,1/2, with $\theta=45^0$

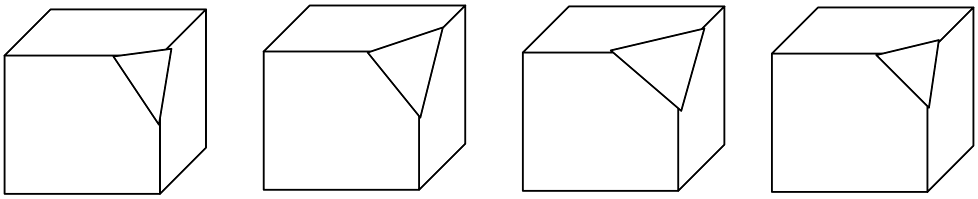


Figure 12: Oblique projections for f=1,7/8,3/4,5/8,1/2, with $\theta=45^0$ (from left to right)

Example4: Find the transformation matrix for a) cavalier projection with $\theta=45^0$, and b) cabinet projection with $\theta=30^0$ c) Draw the projection of unit cube for each transformation.

Solution: We know that cavalier and cabinet projections are a special case of an oblique projection. The transformation matrix for oblique projection is:

$$P_{obligue} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ f.\cos\theta & f.\sin\theta & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(ii) To draw the cabinet projection, we find the image coordinates of a unit cube as:

$$P' \; V.\; Pcab = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ \sqrt{3}/4 & 5/4 & 0 & 1 \\ \sqrt{3}/4 & 1/4 & 0 & 1 \\ (1+\sqrt{3}/4) & 1/4 & 0 & 0 \\ (1+\sqrt{3}/4) & 5/4 & 0 & 1 \end{bmatrix}$$

Hence, the image coordinates are:
A' (0, 0, 0), B' = (1,0,0), C' = (1, 1, 0), D' = (0, 1, 0), E' = (√3/4, 5/4, 0)
F' = (√3/4, 1/4, 0), G' = (1 + √3/4, 1/4, 0), H' = (1 + √3/4, 5/4, 0)

The following *Figure (g)* shows a cabinet projection of a unit cube.

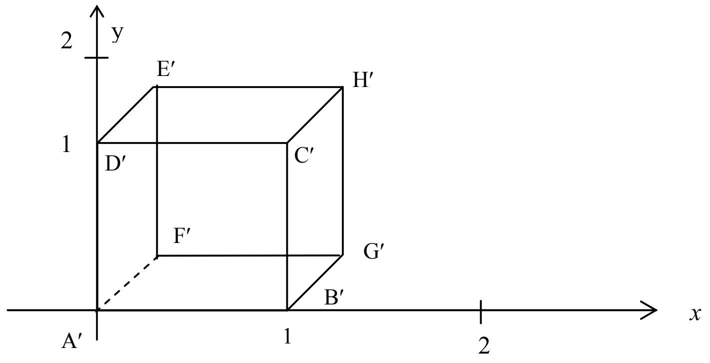


Figure (g)

2.2.1.2 Isometric Projection

There are 3 common sub categories of Orthographic (axonometric) projections:

1) *Isometric*: The direction of projection makes equal angles with all the three principal axes.
2) *Dimetric*: The direction of projection makes equal angles with exactly two of the three principal axes.
3) *Trimetric*: The direction of projection makes unequal angles with all the three principal axes.

Isometric projection is the most frequently used type of *axonometric* projection, which is a method used to show an object in all three dimensions in a single view. Axonometric projection is a form of orthographic projection in which the projectors are always perpendicular to the plane of projection. However, the object itself, rather than the projectors, are at an angle to the plane of projection.

Figure13 shows a cube projected by isometric projection. The cube is angled so that all of its surfaces make the same angle with the plane of projection. As a result, the length of each of the edges shown in the projection is somewhat shorter than the actual length of the edge on the object itself. This reduction is called foreshortening. Since, all of the surfaces make the angle with the plane of projection, the edges foreshorten in the same ratio. Therefore, one scale can be used for the entire layout; hence, the term *isometric* which literally means the same scale.

Construction of an Isometric Projection

In isometric projection, the direction of projection $d = (d_1,d_2,d_3)$ makes an equal angles with all the three principal axes. Let the direction of projection $d = (d_1,d_2,d_3)$ make equal angles (say α) with the positive side of the x,y, and z axes(see *Figure 13*).

Then

$i.d=d_1=|i|.|d|.\cos\alpha \Rightarrow \cos\alpha=d_1/|d|$

similarly

$d_2=j.d=|j|.|d|.\cos\alpha \Rightarrow \cos\alpha=d_2/|d|$

$d_3=k.d=|k|.|d|.\cos\alpha \Rightarrow \cos\alpha=d_3/|d|$

so $\cos\alpha=d_1/|d| = d_2/|d| = d_3/|d|$

⇨ $d_1= d_2 = d_3$ is true

we choose $d_1=d_2=d_3=1$

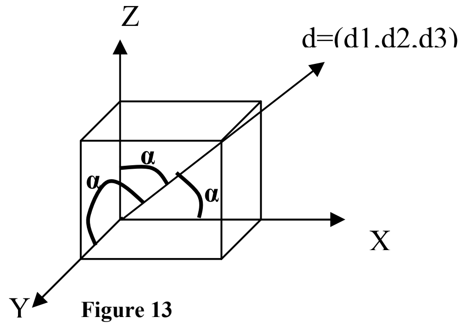


Figure 13

Thus, we have **d =(1, 1, 1)**

Since, the projection, we are looking for is an isometric projection => orthographic projection, i.e, the plane of projection, should be perpendicular to d, so d = n = (1,1,1). Also, we assume that the plane of projection is passing through the origin.

⇨ We know that the equation of a plane passing through reference point $R(x_0,y_0,z_0)$ and having a normal $N = (n_1,n_2,n_3)$ is: $(x - x_0).n_1 + (y - y_0).n_2 + (z - z_0).n_3=0$ ------------------------(14)

Since $(n_1,n_2,n_3)=(1,1,1)$ and

$(x_0,y_0,z_0)=(0,0,0)$

From equation (14), we have $\mathbf{x + y + z = 0}$

Thus, we have the equation of the plane: $\mathbf{x + y + z = 0}$ and **d = (1,1,1)**

Transformation for Isometric projection

Let P(x,y,z) be any point in a space. Suppose a given point P(x,y,z) is projected to P'(x'y',z') onto the projection plane x + y + z = 0. We are interested to find out the projection point P'(x',y',z').

The parametric equation of a line passing through point P(x, y, z) and in the direction of d (1, 1, 1) is:

P + t.**d** = (x, y, z) + t. (1,1,1) = (x + t, y + t, z + t) is any point on the line, where $-\infty< t < \infty$. The point P' can be obtained, when $t = t^*$.

Thus $P'=(x',y',z')=(x + t^*,y + t^*,z + t^*)$, since P' lies on x + y + z = 0 plane.

⇨ $(x + t^*)+(y + t^*) + (z + t^*)=0$

⇨ $3.t^*=-(x + y + z)$

⇨ $t^*=-(x + y + z)/3$ should be true.

⇨ $x'= (2.x - y - z)/3$, $y'=(-x +2.y - z)/3$, $z'=(- x - y +2.z)/3$

Thus, P’=(x’,y’,z’)=[(2.x –y–z)/3, (–x +2.y– z)/3, (-x-y+2.z)/3] --------------------(15)

In terms of homogeneous coordinates, we obtain

$$(x', y', z, 1) = (x, y, z, 1)\begin{pmatrix} 2/3 & -1/3 & 1/3 & 0 \\ -1/3 & 2/3 & -1/3 & 0 \\ -1/3 & -1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note: We can also verify this Isometric transformation matrix by checking all the foreshortening factors, i.e., to check whether all the foreshortening factors (fx, fy, fz) are equal or not. Consider the points A,B and C on the coordinate axes (see *Figure14*).

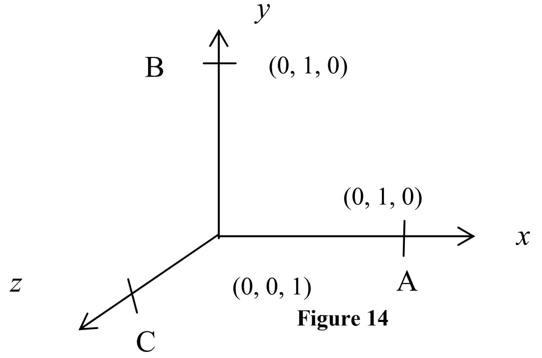


Figure 14

i) Take OA, where O=(0,0,0) and A (1,0,0). Suppose O is projected to O’ and A is projected to A’
   Thus, by using equation (15), we have O’=(0,0,0) and A’=(2/3,-1/3,-1/3).
   So $|O'A'| = \sqrt{(2/3)^2 + (-1/3)^2 + (-1/3)^2} = \sqrt{2/3}$ = fx ------------(16)

ii) Take OB, where O = (0,0,0) and B (0,1,0). Suppose O is projected to O’ and B is projected to B’. Thus by using equation (15), we have O’=(0,0,0) and B’= (-1/3,2/3,-1/3).
   So $|O'B'| = \sqrt{-(1/3)^2 + (2/3)^2 + (-1/3)^2} = \sqrt{2/3}$ = fy -------------(17)

iii) Take OC, where O=(0,0,0) and C(0,0,1). Suppose O is projected to O’ and C is projected to C’

   Thus, by using equation(15), we have O’=(0,0,0) and C’=(-1/3,-1/3,2/3).

   So $|O'C'| = \sqrt{(-1/3)^2 + (-1/3)^2 + (2/3)^2} = \sqrt{2/3}$ =fz --------------(18)

Thus, we have **fx=fy=fz**, which is true for Isometric projection.

Example 5: Obtain the isometric view of a cuboid, shown in *figure*. The size of cuboid is 10x8x6, which is lying at the origin.

Solution: The given cuboids can be represented in terms of Homogeneous coordinates of vertices as shown in *Figure (h)**:***

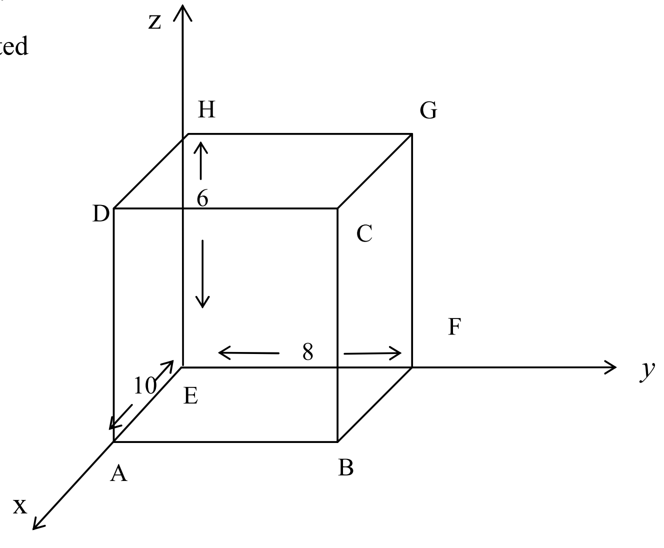


Figure (h)

$$V = [A\ B\ C\ D\ E\ F\ G\ H] = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix} \begin{pmatrix} 10 & 0 & 0 & 1 \\ 10 & 8 & 0 & 1 \\ 10 & 8 & 6 & 1 \\ 10 & 8 & 6 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 8 & 0 & 1 \\ 0 & 8 & 6 & 1 \\ 0 & 0 & 6 & 1 \end{pmatrix}$$

To draw an Isometric projection, we find the image coordinate of a given cuboid as follows:

$$P' = V.\ P_{ISO} = \begin{pmatrix} 10 & 0 & 0 & 1 \\ 10 & 8 & 0 & 1 \\ 10 & 8 & 6 & 1 \\ 10 & 0 & 6 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 8 & 0 & 1 \\ 0 & 8 & 6 & 1 \\ 0 & 0 & 6 & 1 \end{pmatrix} . \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} =$$

$$\begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{bmatrix} 20 & -10 & -10 & 3 \\ 12 & 8 & -18 & 3 \\ 6 & 0 & -6 & 3 \\ 14 & -16 & 2 & 3 \\ 0 & 0 & 0 & 3 \\ -8 & 16 & -8 & 3 \\ -14 & 10 & 4 & 3 \\ -6 & -6 & 12 & 3 \end{bmatrix} = \begin{bmatrix} 6.66 & -3.33 & -3.33 & 1 \\ 4.0 & 2.66 & -6.0 & 1 \\ 2 & 0 & -2.0 & 1 \\ 4.66 & -5.33 & 0.66` & 1 \\ 0 & 0 & 0 & 1 \\ -2.66 & 5.33 & 1.33 & 1 \\ -4.66 & 3.33 & 1.33 & 1 \\ -2.0 & -2.0 & 4.0 & 1 \end{bmatrix}$$

Thus, by using this matrix, we can draw an isometric view of a given cuboids.

☞ Check Your Progress 2

1) When all the foreshortening factors are different, we have
   a) Isometric b) Diametric c) Trimetric Projection d) All of these.

   ………………………………………………………………………………………

   ………………………………………………………………………………………

   ………………………………………………………………………………………

   ………………………………………………………………………………………

2) Distinguish between Orthographic and Oblique parallel projection.

   ………………………………………………………………………………………

   ………………………………………………………………………………………

   ………………………………………………………………………………………

   ………………………………………………………………………………………

3) What do you mean by foreshortening factor. Explain Isometric, Diametric and Trimetric projection using foreshortening factors.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

4) Show that for Isometric projection the foreshortening factor along x, y and z-axes must be $\sqrt{2/3}$, i.e. fx = fy = fz = $\sqrt{2/3}$

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

5) Consider a parallel projection with the plane of projection having the normal (1,0,–1) and passing through the origin O(0,0,0) and having a direction of projection **d** = (–1,0,0). Is it orthographic projection? Explain your answer with reason.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

6) Compute the cavalier and cabinet projections with angles of 45^0 and 30^0 respectively of a pyramid with a square base of side 4 positioned at the origin in the xy-plane with a height of 10 along the z-axis.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

2.2.2 Perspective Projections

In a perspective projection the center of projection is at finite distance. This projection is called perspective projection because in this projection faraway objects look small and nearer objects look bigger. *See Figure 15 and 16.*

In general, the plane of projection is taken as Z=0 plane.

Properties

1) Faraway objects look smaller.
2) Straight lines are projected to straight lines.
3) Let l_1 and l_2 be two straight lines parallel to each other. If l_1 and l_2 are also parallel to the plane of projection, then the projections of l_1 and l_2 (call them l'_1 and l'_2), will also be parallel to each other.
4) If l_1 and l_2 be two straight lines parallel to each other, but are not parallel to the plane of projection, then the projections of l_1 and l_2 (call them l'_1 and l'_2), will meet in the plane of projection (see *Figure 16*).

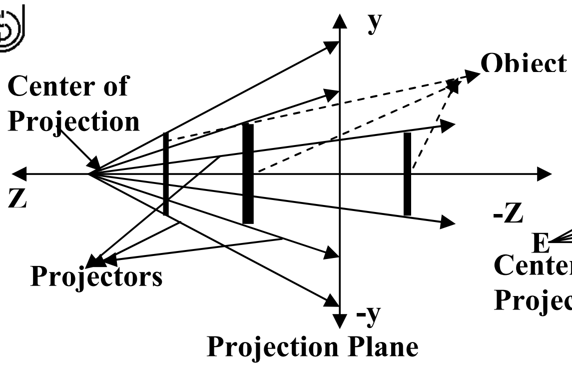


Figure 15

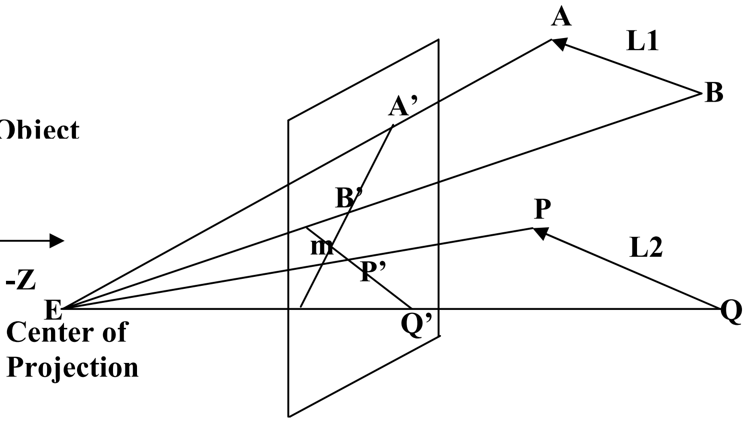


Figure 16

The infinite lines AB and PQ will be projected to the lines A'B' and P'Q' respectively on the plane of projection. That is all points of the line AB is projected to all points of the line A'B'. Similarly all points of the line PQ is projected to all points of the line P'Q'. But A'B' and P'Q' intersect at M and M is the projection of some point on the line AB as well as on PQ, but AB ‖ PQ, which implies that M is the projection of point at infinity where AB and PQ meet. In this case M is called a **Vanishing point.**

Principle Vanishing point

Suppose l_1 and l_2 be two straight lines parallel to each other, which are also parallel to x-axis. If the projection of l_1 and l_2 (call them l'_1 and l'_2), appears to meet at a point (point at infinity), then the point is called a Principle vanishing point w.r.t. the x-axis. Similarly we have Principle vanishing point w.r.t. the y-axis and z-axis.

Remark

A Perspective projection can have at most 3 Principle Vanishing points and at least one Principle vanishing point.

To understand the effects of a perspective transformation, consider the *Figure 17.* This *figure* shows the perspective transformation on z=0 plane of a given line AB which is parallel to the z-axis. The A*B* is the projected line of the given line AB in the z=0 plane. Let a centre of projection be at (0,0,-d) on the z-axis. The *Figure (A)* shows that the line A'B' intersects the z=0 plane at the same point as the line AB. It also intersects the z-axis at z=+d. It means the perspective transformation has transformed the intersection point.

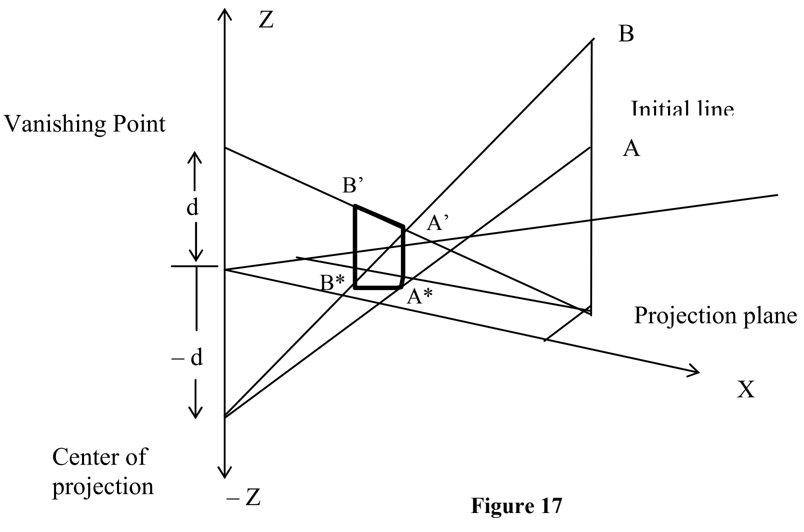


Figure 17

Mathematical description of a Perspective Projection

A perspective transformation is determined by prescribing a C.O.P. and a viewing plane. Let P(x,y,z) be any object point in 3D and C.O.P. is at E(0,0,-d). The problem is to determine the image point coordinates P'(x',y',z') on the Z=0 plane (see *Figure 18*).

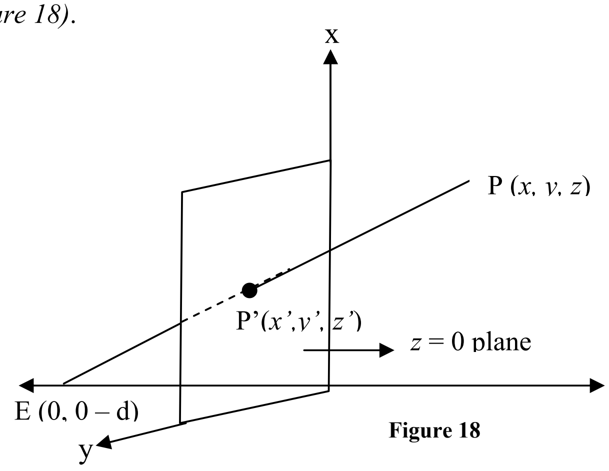


Figure 18

The parametric equation of a line EP, starting from E and passing through P is:

$E+t(P-E)\quad 0<t<\infty$

$=(0,0,-d)+t[(x,y,z)-(0,0,-d)]$

$=(0,0,-d)+t(x,y,z+d)$

$=[t.x,\ t.y,\ -d+t.(z+d)]$

Point P' is obtained, when t=t*

That is, $P'=(x',y',z')=[t^*.x,\ t^*.y,\ -d+t^*.(z+d)]$

Since P' lies on Z=0 plane implies $-d+t^*.(z+d)=0$ must be true, that is $t^*=d/(z+d)$ is true.

Thus $x'=t^*.x=x.d/(z+d)$
$y'=t^*.y=y.d/(z+d)$
$z'=-d+t^*(z+d)=0$

thus $P'=(\ x.d/(z+d),\ y.d/(z+d),\ 0)$
$=(x/((z/d)+1),y/((z/d)+1),0)$
in terms of Homogeneous coordinate system $P'=(x,y,0,(z/d)+1)$. ---------(5)

The above equation can be written in matrix form as:

$$P(x',y',z',1)=(x,y,z,1)\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1/d\\0&0&0&1\end{pmatrix} = [x,y,0,(z/d)+1] \qquad \text{-----------(1)}$$

That is, $P'_h = P_h.P_{per,z}$ ----------------- (2)

Where $P_{per,z}$ in equation (4.6) represents the *single point perspective transformation on z-axis*.

The Ordinary coordinates are:
$[x',y',z',1]=[x/(r.z+1),y/(r.z+1),0,1]$ where $r=1/d$ ---------------- (3)

Vanishing Point

The vanishing point is that point at which parallel lines appear to converge and vanish. A practical example is a long straight railroad track.

To illustrate this concept, consider the *Figure 17* which shows a perspective transformation onto z=0 plane. The *Figure17* shows a Projected line A*B* of given line AB parallel to the z-axis. The center of projection is at (0,0,-d) and z=0 be the projection plane.

Consider the perspective transformation of the point at infinity on the +z-axis, i.e.,

$$[0,0,1,0] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{pmatrix} = (0,0,0,1/d) \text{ -------------- } (4)$$

Thus, the ordinary coordinates of a point (x',y',z',1)=(0,0,0,1), corresponding to the transformed point at infinity on the z-axis, is now a finite point. This means that the entire semi-infinite positive space(0<=z<=∞) is transformed to the finite positive half space 0<=z'<=d.

Single point perspective transformation

In order to derive the single point perspective transformations along x and y-axes, we construct *Figures (19) and (20)* similar to *Figure 18*, but with the corresponding COP's at E(-d,0,0) and E(0,-d,0) on the negative x and y-axes respectively.

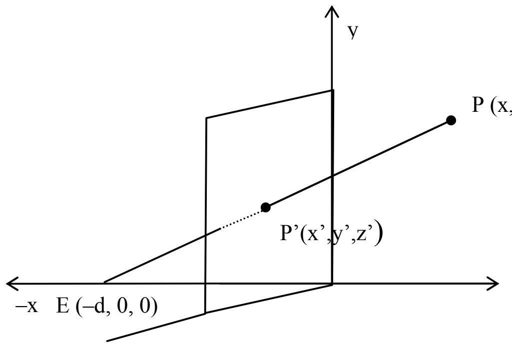


Figure 19

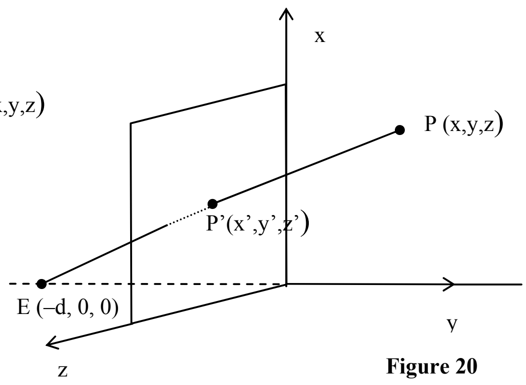


Figure 20

The parametric equation of a line EP, starting from E and passing through P is:

E+t(P-E) 0<t<∞

=(-d,0,0)+t[(x,y,z)-(-d,0,0)]

=(-d,0,0)+t[x+d,y,z]

=[-d+t.(x+d), t.y, t.z]

Point P' is obtained, when t=t*

That is, P'=(x',y',z') =[-d+t*.(x+d), t*.y, t*.z]

Since, P' lies on X=0 plane implies -d+t*.(x+d)=0 must be true, that is t*=d/(x+d) is true.

Thus, x'=-d+t*(x+d)=0
y'=t*.y=y.d/(x+d)
z'=t*.z=z.d/(x+d)
thus P'=(0, y.d/(z+d), z.d/(x+d))
=(0,y/((z/d)+1), z/((x/d)+1))

in terms of Homogeneous coordinate system P'=(0,y,z,(x/d)+1).

The above equation can be written in matrix form as:

$$P(x',y',z',1)=(x,y,z,1)\begin{pmatrix}0 & 0 & 0 & 1/d\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} =[0,y,z,(x/d)+1]$$

$$=[0,y/((z/d)+1),\ z/((x/d)+1),1] \text{ -------- (5)}$$

That is, $P'_h = P_h.P_{per,x}$ --------------------------------(6)

Where $P_{per,z}$ in equation (5) represents the *single point perspective transformation* w.r.t. x-axis.

Thus, the ordinary coordinates(projected point P' of a given point P) of a *single point perspective transformation* w.r.t. x-axis is:

(x',y',z',1)= [0,y/((z/d)+1), z/((x/d)+1),1] has a center of projection at [-d,0,0,1] and a vanishing point located on the x-axis at [0,0,0,1]

Similarly, the *single point perspective transformation w.r.t. y-axis* is therefore:

$$P(x',y',z',1)=(x,y,z,1)\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1/d\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} =[x,0,z,(y/d)+1]$$

$$=[x/((y/d)+1),0,\ z/((y/d)+1),1]$$

That is, $P'_h = P_h.P_{per,y}$ ----------------------------(7)

Where $P_{per,y}$ in equation (5) represents the *single point perspective transformation* w.r.t. y-axis.

Thus, the ordinary coordinates(projected point P' of a given point P) of a *single point perspective transformation* w.r.t. y-axis is:

(x',y',z',1)=[x/((y/d)+1),0, z/((y/d)+1),1] has a center of projection at [0,-d,0,1] and a vanishing point located on the y-axis at [0,0,0,1].

Example 6: Obtain a transformation matrix for perspective projection for a given object projected onto x=3 plane as viewed from (5,0,0).

Solution: Plane of projection: $x = 3$ (given)
Let P (x, y, z) be any point in the space. We know the
Parametric equation of a line AB, starting from A and passing

through B is

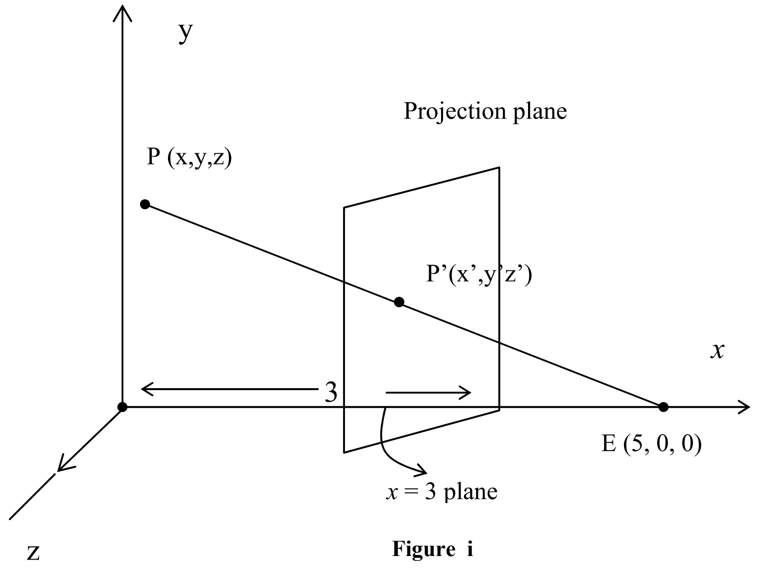


Figure i

P (t) = A + t. (B – A), o < t < ∞

So that parametric equation of a line starting from E (5,0,0) and passing through P (x, y, z) is:

E + t (P – E) , o < t < ∞.
= (5, 0, 0) + t [(*x, y, z*) – (5, 0, 0)]
= (5, 0, 0) + [t (*x* – 5), t. *y*, t. *z*]
= [t. (x – 5) + 5, t. y, t. z]. Assume

Point P' is obtained, when t = t*

$\therefore$ P' = (*x*', *y*', *z*') = [t* (*x* – 5) + 5, t**y*, t*. *z*]

Since, P' lies on *x* = 3 plane, so
t* (*x* – 5) + 5 = 3 must be true;

$$t^* = \frac{-2}{x-5}$$

$$P' = (x', y', z') = \left(3, \frac{-2.y}{x-5}, \frac{-2.z}{x-5}\right)$$

$$= \left(\frac{3x-15}{x-5}, \frac{-2.y}{x-5}, \frac{-2.z}{x-5}\right)$$

In Homogeneous coordinate system

$$P' = (x', y', z', 1) = \left(\frac{3x-15}{x-5}, \frac{-2.y}{x-5}, \frac{2.z}{x-5}, 1\right)$$

$$= (3x - 15, -2.y, -2.z, x - 5) \qquad \text{--------------(1)}$$

In Matrix form:

$$(x', y', z', 1) = (x, y, z, 1)\begin{bmatrix} 3 & 0 & 0 & 1 \\ 0 & -2 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ -15 & 0 & 0 & -5 \end{bmatrix} \text{-------------- (2)}$$

Thus, equation (2) is the required transformation matrix for perspective view from (5, 0, 0).

Example 7: Consider the line segment AB in 3D parallel to the z-axis with end points A (– 5,4,2) and B (5,-6,18). Perform a perspective projection on the X=0 plane, where the eye is placed at (10,0,10).

Solution: Let P (x, y, z) be any point in the space.

The parametric equation of a line starting from E and passing through P is:

$$E + t.\,(P - E),\ o < t < 1.$$
$$= (10,0,10) + t.\,[(x, y, z) - (10, 0, 10)]$$
$$= (10, 0,10) + t\,[(x - 10)], y\,(z - 10)]$$
$$= (t.\,(x - 10) + 10,\ t.\,y,\ t\,(z - 10) + 10)$$

Assume point P' can be obtained, when t = t*

$\therefore$ P' = (x', y', z') = (t* (x – 10) + 10, t*.y, t*. (z – 10) + 10)

since point P' lies on x = 0 plane

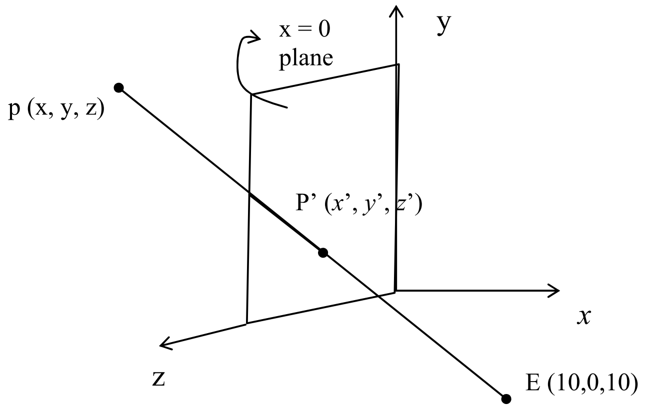


Figure j

$$= t^*\,(x - 10) + 10 = 0$$
$$= t^* = \frac{-10}{x-10}$$
$$= P' = (x', y', z') = \left(0, \frac{-10.y}{x-10}, \frac{-10(z-10)}{x-10} + 10\right)$$

$$\left(0, \frac{-10.y}{x-10}, \frac{10.x-10.z}{x-10}\right)$$

In terms of Homogeneous coordinate system;

$$P' = (x', y', z', 1) = \left(0, \frac{-y}{\frac{x}{10}-1}, \frac{x-z}{\frac{x}{10}-1}, 1\right) = (0, -y, x-z, \frac{x}{10} - 1)$$

In Matrix form

$$(x', y', z', 1) = (x, y, z, 1)\begin{bmatrix} 0 & 0 & 1 & 1/10 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \text{------------------(1)}$$

This equation (1) is the required perspective transformation, which gives a coordinates of a projected point P' (x', y', z') onto the $x = 0$ plane, when a point p (x, y, z) is viewed from E (10, 0, 10)

Now, for the given points A (–5, 4, 2) and B (5, –6, 18), A' and B' are their projection on the $x = 0$ plane.

Then from Equation (1).

$$A' = (x'_1, y'_1, z'_1, 1) = (-5, 4, 2, 1).\begin{bmatrix} 0 & 0 & 1 & 1/10 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

$$= \quad (0, -4, -7, \frac{-5}{10} - 1)$$

$$= (0, -4, -7, \frac{-15}{10})$$

$$= (0, -40, -70, -15)$$

$$= (0, \frac{40}{15}, \frac{70}{15}, 1)$$

Hence $x_1' = 0$; $y_1' = 2.67$; $z_1' = 4.67$

similarly $B' = (x_2', y_2', z_2', 1) = (5, -6, 18, 1) . \begin{bmatrix} 0 & 0 & 1 & 1/10 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$

$$= (0, 60, -130, -5)$$

$$= (0, -12, 26, 1)$$

Hence $x_2' = 0$; $y_2' = -12$; $z_2' = 26$

Thus the projected points A' and B' of a given points A and B are:

$A' = (x_1', y_1' z_1') = (0, 2.67, 4.67)$ and $B' = (x_2', y_2', z_2') = (0, -12, 26, 1)$

Example 8: Consider the line segment AB in *Figure k*, parallel to the z-axis with end points A (3, 2, 4) and B (3, 2, 8). Perform a perspective projection onto the z = 0 plane from the center of projection at E (0, 0, – 2). Also find out the vanishing point.

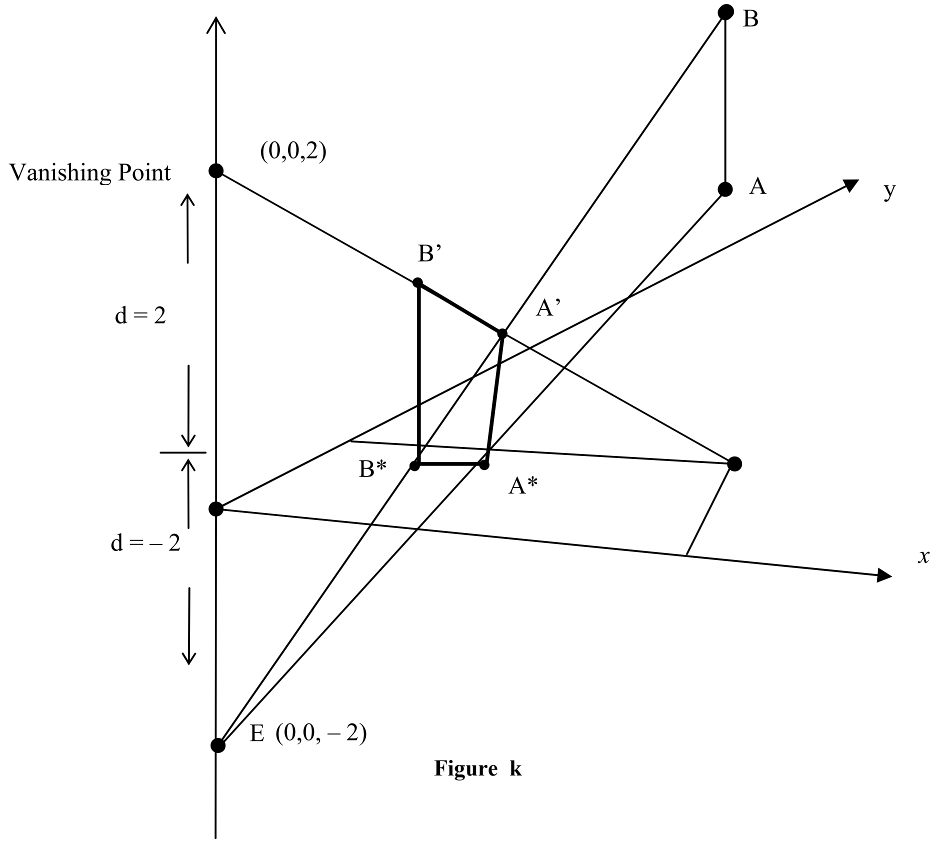


Figure k

Solution. We know that (from Equation (1)), the center of single point perspective transformation: of a point P (x, y, z) onto z = 0 plane, where center of projection is at (0, 0, – d) is given by:

$$(x', y', z', 1) = (x, y, z, 1) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P'n = Pn.\ P_{per,z} \quad \text{-----------------------(I)}$$

Thus the perspective transformation of a given line AB to A* B* with d = 2 is given by:

$$V'_n = V_n.\ P_{per,z}$$

$$\begin{matrix} A^* \\ B^* \end{matrix} \begin{bmatrix} x_1^* & y_1^* & z_1^* & 1 \\ x_1^* & y_2^* & z_1^* & 1 \end{bmatrix} = \begin{matrix} A \\ B \end{matrix} \begin{bmatrix} 3 & 2 & 4 & 1 \\ 3 & 2 & 8 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{matrix} A^* \\ B^* \end{matrix} \begin{bmatrix} 1 & 0.667 & 0 & 1 \\ 0.6 & 0.4 & 0 & 1 \end{bmatrix}$$

Hence, the projected points of a given line AB is:

A* = (1, 0.667, 0)
B* = (0.6, 0.4, 0)

The vanishing point is (0, 0, 0).

Example 9: Perform a perspective projection onto the z = 0 plane of the unit cube, shown in *Figure (l)* from the cop at E (0, 0, 10) on the z-axis.

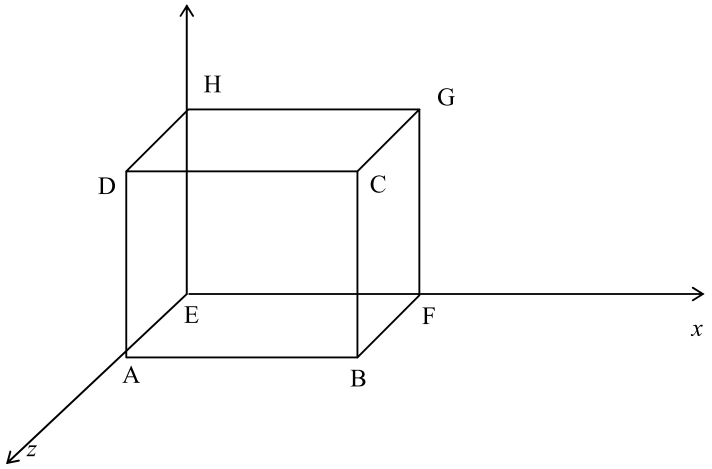


Figure (l)

01: Here center of projection

$E = (0, 0, -d) = (0, 0, 10).$

$\therefore\ d = -10$

we know that (from equation – 1), the single point perspective transformation of the projection with $z = 0$, plane, where cop is at (0, 0, –d) is given by:

$$(x', y', z', 1) = (x, y, z, 1)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{-----------(I)}$$

$$P_n' = P.\ P_{per,\,z} \quad \text{--------------- (II)}$$

Thus the perspective transformation of a given cube v = [ABCDEFGH] to V' = [A'B'C'D'E'F'G'H'] with d = – 10 is given by:

$$[V'] = [V]\ .\ [P_{per,\,z}]$$

$$= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$V' = \begin{bmatrix} 0 & 0 & 0 & 0.9 \\ 1 & 0 & 0 & 0.9 \\ 1 & 1 & 0 & 0.9 \\ 0 & 1 & 0 & 0.9 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1.11 & 0 & 0 & 1 \\ 1.11 & 1.11 & 0 & 1 \\ 0 & 1.11 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Thus the projected points of a given cube V = [ABCDEFGH] are:
A' =(0, 0, 0), B' = (1.11, 0, 0), C' = (1.11, 1.11, 0), D' = (0, 1.11, 0), E' = (0, 0, 0)
F' = (1, 0, 0), G' = (1, 1, 0) and H' = (0, 1, 0).

☞ Check Your Progress 3

1) Obtain the perspective transformation onto z = d plane, where the c. o. p. is at the origin.

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

2) Consider a cube given in example – 4, the cube can be centered on the z-axis by translating it $-\frac{1}{2}$ units in the x y directions perform a single point perspective transformation onto the z = 0 plane, with c. o. p. at Zc = 10 on the z-axis.

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

3) A unit cube is placed at the origin such that its 3-edges are lying along the x, y and z-axes. The cube is rotated about the y-axis by 30°. Obtain the perspective projection of the cube viewed from (80, 0, 60) on the z = 0 plane.

……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

Two-Point and Three-Point Perspective transformations

The 2-point perspective projection can be obtained by rotating about one of the principal axis only and projecting on X=0 (or Y=0 or Z=0) plane. To discuss the phenomenon practically consider an example for 3-point perspective projection (given below) some can be done for 2-point aspect.

Example 10: Find the principal vanishing points, when the object is first rotated w.r.t. the y-axis by – 30° and *x*-axis by 45°, and projected onto z = 0 plane, with the center of projection being (0, 0, – 5).

Solution: Rotation about the y-axis with angle of rotation

$\theta = (-30°)$ is

$$[R_y] = [Ry]_{\theta=-30} = \begin{bmatrix} \cos(30°) & 0 & -\sin(-30°) \\ 0 & 1 & 0 \\ \sin(-30°) & 0 & \cos(-30°) \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ -1/2 & 0 & \sqrt{3}/2 \end{bmatrix}$$

Similarly Rotation about the x-axis with angle of Rotation ϕ 45° is:

$$[R_x] = [R_x]_{45°} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\therefore\ [R_y].[R_x] = \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ -1/2 & 0 & \sqrt{3}/2 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 & -1/2\sqrt{2} & 1/2\sqrt{2} \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ -1/2 & -3/2\sqrt{2} & \sqrt{3}/2\sqrt{2} \end{bmatrix} \text{-----------------(1)}$$

Projection: Center of projection is E (0, 0, – 5) and plane of projection is z = 0 plane.

For any point p (x, y, z) from the object, the Equation of the ray starting from E and passing through the point P is:

$$E + t\,(P - E),\ t > 0$$

i.e. $(0, 0, -5) + t\,[(x, y, z) - (0, 0, -5)]$

$= (0, 0, -5) + t\,(x, y, z + 5)$

$= (t.x,\ t.\ y,\ -5 + t\,(z + 5)$

for this point to be lie on z = 0 plane, we have:

$-5 + t\,(z + 5) = 0$

$\therefore\ t = \dfrac{5}{z+5}$

$\therefore$ the projection point of p (x, y, z) will be:

$$P' = (x', y', z') = \left(\frac{5.x}{z+5}, \frac{5.y}{z+5}, 0\right)$$

In terms of homogeneous coordinates, the projection matrix will become:

$$[P] = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 5 \end{bmatrix} \text{-------------(2)}$$

$$\therefore [R_y].\ [R_x].[P] = \begin{bmatrix} \sqrt{3}/2 & -1/2\sqrt{2} & 1/2\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1\sqrt{2} & -\sqrt{3}/2\sqrt{2} & \sqrt{3}/2\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{5\sqrt{3}}{2} & \frac{-5}{2\sqrt{2}} & 0 & \frac{1}{2\sqrt{2}} \\ 0 & \frac{5}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{-5}{2} & \frac{-5\sqrt{3}}{2\sqrt{2}} & 0 & \frac{\sqrt{3}}{2\sqrt{2}} \\ 0 & 0 & 0 & 5 \end{bmatrix} \qquad \text{----------------(3)}$$

Let (x, y, z) be projected, under the combined transformation (3) to (x', y', z'), then

$$(x', y', z', 1) = (x, y, z, 1) \begin{bmatrix} \frac{5\sqrt{3}}{2} & \frac{-5}{2\sqrt{2}} & 0 & \frac{1}{2\sqrt{2}} \\ 0 & \frac{5}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{-5}{2} & \frac{-5\sqrt{3}}{2\sqrt{2}} & 0 & \frac{\sqrt{3}}{2\sqrt{2}} \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

$$= x' = \frac{\left(\frac{5\sqrt{3}}{2}.x - \frac{5}{2}.z\right)}{\left(\frac{x}{2\sqrt{2}} + \frac{y}{\sqrt{2}} + \frac{\sqrt{3}.z}{2\sqrt{2}} + 5\right)}$$

and

$$y' = \left.\frac{\left(\frac{-5}{2\sqrt{2}}.x + \frac{5}{\sqrt{2}}.y - \frac{5\sqrt{3}}{2\sqrt{2}}.z\right)}{\left(\frac{x}{2\sqrt{2}} + \frac{y}{\sqrt{2}} + \frac{\sqrt{3}}{2\sqrt{2}}.z + 5\right)}\right\} \text{-----------(4)}$$

Case 1: Principal vanishing point w.r.t the x-axis.

By considering first row of the matrix (Equation – (3)), we can claim that the principal vanishing point (w.r.t) the x-axis) will be:

$$\left(\frac{\frac{5\sqrt{3}}{2}}{\frac{1}{2\sqrt{2}}}, \frac{\frac{-5}{2\sqrt{2}}}{\frac{1}{2\sqrt{2}}}, 0 \right)$$

i.e., $(5\sqrt{6}, -5, 0)$ -------------------(I)

In order to varify our claim, consider the line segments AB, CD, which are parallel to the x-axis, where A = (0, 0, 0), B = (1, 0, 0), C = (1, 1, 0), D = (0, 1, 0)

If A', B', C', D' are the projections of A, B, C, D, respectively, under the projection matrix (3), then

$$A' = (0, 0, 0),\ B' = \left(\frac{5\sqrt{3}}{\frac{1}{2\sqrt{2}} + 5}, \frac{\frac{-5}{2\sqrt{2}}}{\frac{1}{2\sqrt{2}} + 5}, 0 \right)$$

$$C' = \left(\frac{\frac{5\sqrt{3}}{2}}{\left(\frac{1}{2\sqrt{2}} + \frac{1}{\sqrt{2}} + 5\right)}, \frac{\left(-\frac{5}{2\sqrt{2}} + \frac{5}{\sqrt{2}}\right)}{\left(\frac{1}{2\sqrt{2}} + \frac{1}{\sqrt{2}} + 5\right)}, 0 \right)$$

$$D' = \left(0, \frac{5/\sqrt{2}}{\left(\frac{1}{\sqrt{2}} + 5\right)}, 0 \right) \qquad \text{\{Using Equation (4)\}}$$

$$A' = (0,0,0)\ ,\ B' = \left(\frac{5\sqrt{6}}{1+10\sqrt{2}}, \frac{-5}{1+10\sqrt{2}}, 0 \right),$$

$$C' = \left(\frac{5\sqrt{6}}{3+10\sqrt{2}}, \frac{5}{3+10\sqrt{2}}, 0 \right) \text{ and}$$

$$D' = \left(0, \frac{5}{1+5\sqrt{2}}, 0 \right)$$

Consider the line equation of A’B’: The parametric Equation is:

A’ + t (B’ – A’)

i.e. $(0, 0, 0) + t \left(\frac{5\sqrt{6}}{1+10\sqrt{2}}, \frac{-5}{1+10\sqrt{2}}, 0 \right)$

$$= \left(\frac{5t\sqrt{6}}{1+10\sqrt{2}}, \frac{-5.t}{1+10\sqrt{2}}, 0 \right)$$

we will verify that the vanishing point (I) lies on this line:

i.e. $\left(\frac{5t\sqrt{6}}{1+10\sqrt{2}}, \frac{-5.t}{1+10\sqrt{2}}, 0 \right) = (5\sqrt{6}, -5, 0)$

$$= \frac{5.t.\sqrt{6}}{1+10\sqrt{2}} = 5\sqrt{6}$$

and $\frac{-5t}{1+10\sqrt{2}} = -5$ -----------------(5)

must be true for some ‘t’ value.

$t = (1 + 10\sqrt{2}\,)$

then the equation (5) is true and hence (I) lies on the line A’B’.
Similarly consider the line equation C’D’: The parametric Equation is:

C’ + s (D’ – C’) i.e.

$$= \left(\frac{5\sqrt{6}}{3+10\sqrt{2}}, \frac{5}{3+10\sqrt{2}}, 0\right) + s\left[\left(0, \frac{5}{1+5\sqrt{2}}, 0\right) - \left(\frac{5\sqrt{6}}{3+10\sqrt{2}}, \frac{5}{3+10\sqrt{2}} 0\right)\right]$$

$$= \left(\frac{5\sqrt{6}}{3+10\sqrt{2}}, \frac{5}{3+10\sqrt{2}}, 0\right) + s\left(\frac{-5\sqrt{6}}{3+10\sqrt{2}}, \frac{5}{1+5\sqrt{2}} - \frac{5}{3+10\sqrt{2}}, 0\right) \text{ and}$$

$$= \left(\frac{5\sqrt{6}}{3+10\sqrt{2}} \frac{-5.s.\sqrt{6}}{3+10\sqrt{2}}, \frac{5}{3+10\sqrt{2}} + \frac{5.s.(2+5\sqrt{2})}{(1+5\sqrt{2})(3+10\sqrt{2})}, 0\right), \text{ but}$$

we have to verify that the vanishing point (I) lies on C'D'.

i.e. we have to show

$$\left(\frac{5\sqrt{6}}{3+10\sqrt{2}}(1-s) \frac{5}{3+10\sqrt{2}} \left(1+\frac{s(2+5\sqrt{2})}{(1+5\sqrt{2})}\right), 0\right) = (5\sqrt{6}, -5, 0)$$

for some 's' value This holds true if

$$\left.\begin{array}{l} \frac{5\sqrt{6}}{3+10\sqrt{2}}(1-s) = 5\sqrt{6} \\ \text{and } \frac{5}{3+10\sqrt{2}}\left(1+\frac{s(2+5\sqrt{2})}{(1+5\sqrt{2})}\right) = -5 \end{array}\right\} \qquad \text{------------(6)}$$

must holds simultaneously for some 's' value.

If we choose s = –2 (1 + 5 $\sqrt{2}$), then both the conditions of (6) satisfied
$\therefore$ (5 $\sqrt{6}$, –5, 0) lies on C'D'

= (5 $\sqrt{6}$, – 5, 0) is the point at intersection of A'B' and C'D'.
(5 $\sqrt{6}$, – 5, 0) is the principal vanishing point w.r.t. the x-axis.

Case 2: Principal vanishing point w.r.t y-axis:-

From the 2nd Row of the matrix (Equation (3)), the principal vanishing point w.r.t y-axis will be:

$$\left(0, \frac{5}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right) \text{ in homogeneous system.}$$

The vanishing point in Cartesian system is:

$$\left(0, \frac{5/\sqrt{2}}{1/\sqrt{2}}, 0\right) = (0, 5, 0) \qquad \text{------------------------(II)}$$

similar proof can be made to verify our claim:

Case 3: Principal vanishing point w.r.t z-axis:

From the 3rd row of matrix equation (3), we claim that the principal vanishing point w.r.t z-axis will be: $\left(-\frac{5}{2}, \frac{-5\sqrt{3}}{2\sqrt{2}}, 0, \frac{\sqrt{3}}{2\sqrt{2}}\right)$ in Homogeneous system.

In Cartesian system, the vanishing point is:

$$\left(\frac{(-5/2)}{\frac{\sqrt{3}}{2\sqrt{2}}}, \frac{\left(\frac{-5}{2}\frac{\sqrt{3}}{\sqrt{2}}\right)}{\left(\frac{\sqrt{3}}{2\sqrt{2}}\right)}, 0 \right) = \left(\frac{-5\sqrt{2}}{\sqrt{3}}, -5,\ 0 \right) \qquad \text{------------------(III)}$$

A similar proof can be made to verify (III)

General Perspective transformation with COP at the origin

Let the given point P(x,y,z) be projected as P'(x',y',z') onto the plane of projection. The COP is at the origin, denoted by O(0,0,0). Suppose the plane of projection defined by the normal vector **N**=n_1**I**+n_2**J**+n_3**K** and passing through the reference point $R_0(x_0,y_0,z_0)$. From *Figure 21*, the vectors **PO** and **P'O** have the same direction. The vector **P'O** is a factor of **PO**. Hence they are related by the equation: **P'O** = α **PO**, comparing components we have x'=α.x y'=α.y z'=α.z we now find the value of α.

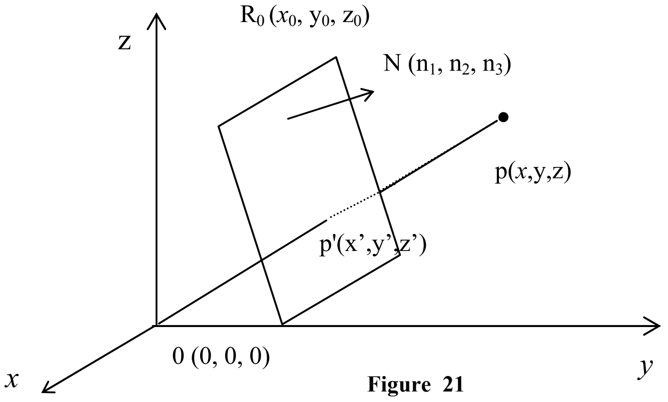


Figure 21

We know that the equation of the projection plane passing through a reference point R_0 and having a normal vector **N**=n_1**I**+n_2**J**+n_3**K** is given by **PR₀.N**=0, that is

$(x-x_0, y-y_0, z-z_0).(n_1, n_2, n_3)=0$ i.e. $n_1.(x-x_0)+n_2.(y-y_0)+n_3.(z-z_0)=0$ ---------( )

since P'(x',y',z') lies on this plane, thus we have: $n_1.(x'-x_0)+n_2.(y'-y_0)+n_3.(z'-z_0)=0$
After substituting x'=α.x ; y'=α.y ; z'=α.z, we have :

$$\alpha = (n_1.x_0+n_2.y_0+n_3.z_0)/(n_1.x+n_2.y+n_3.z) = d_0/(n_1.x+n_2.y+n_3.z)$$

This projection transformation cannot be represented as a 3x3 matrix transformation. However, by using the homogeneous coordinate representation for 3D, we can write this projection transformation as:

$$P_{per,N,Ro} = \begin{pmatrix} d_0 & 0 & 0 & n_1 \\ 0 & d_0 & 0 & n_2 \\ 0 & 0 & d_0 & n_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Thus, the projected point $P'_h(x',y',z',1)$ of given point $P_h(x, y, z, 1)$ can be obtained as

$$P'h = Ph.\ Pper,N,\ Ro = [x, y, z, 1]\begin{pmatrix} d_0 & 0 & 0 & n_1 \\ 0 & d_0 & 0 & n_2 \\ 0 & 0 & d_0 & n_3 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{-----------------(16)}$$

$= [d_0.x,\ d_0.y,\ d_0z,\ (n_1.x + n_2.y + n_3.z)]$

Where $d_0 = n_1.x_0 + n_2.y_0 + n_3.\ z_0$.

General Perspective transformation w.r.t. an arbitrary COP

Let the COP is at C(a,b,c), as shown in *Figure 22*.

From *Figure 7*, the vectors **CP** and **CP'** have the same direction. The vector **CP'** is a factor of **CP**, that is **CP'**=α.**CP**

Thus, (x'-a)= α.(x-a) z
(y'-b)= α.(y-b)
(z'-c)= α.(z-c) -------------(17)

$N = (n_1, n_2, n_3)$

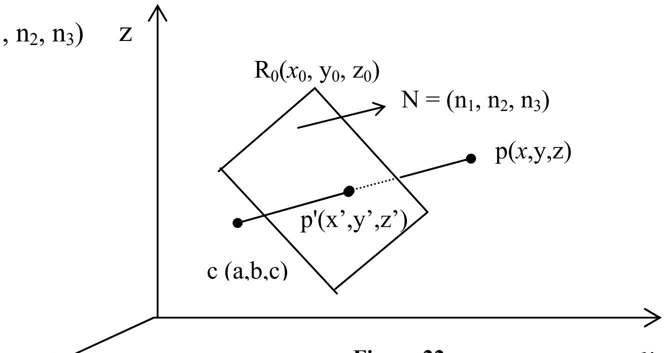


Figure 22

We know that the projection plane passing through a reference point $R_0(x_0,y_0,z_0)$ and having a normal vector N= n1I+n2J+n3K, satisfies the following equation:

n1.(x-x0)+n2.(y-y0)+n3.(z-z0)=0

Since P'(x',y',z') lies on this plane, we have:

n1.(x'-x0)+n2.(y'-y0)+n3.(z'-z0)=0

Substituting the value of x', y' and z', we have:

α= (n1.(x0-a)+n2.(y0-b)+n3.(z0-c))/(n1.(x-a)+n2.(y-b)+n3.(z-c))
=((n1.x0+n2.y0+n3.z0)-(n1.a+n2.b+n3.c))/(n1.(x-a)+n2.(y-b)+n3.(z-c))
=(d0-d1)/(n1.(x-a)+n2.(y-b)+n3.(z-c))
=d/(n1.(x-a)+n2.(y-b)+n3.(z-c))

Here, d=d0-d1= (n1.x0+n2.y0+n3.z0)- (n1.a+n2.b+n3.c) represents perpendicular distance from COP, C to the projection plane.

In order to find out the general perspective transformation matrix, we have to proceed as follows:

Translate COP, C(a,b,c) to the origin. Now, R'0=(x0-a,y0-b,z0-c) becomes the reference point of the translated plane.(but Normal vector will remain same).

Apply the general perspective transformation $P_{per,N,R'o}$

Translate the origin back to C.

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{pmatrix} \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ n_1 & n_2 & n_3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{pmatrix}$$

$$= \begin{pmatrix} d + n_1.a & n_1.b & n_1.c & n_1 \\ n_2.a & d + n_2.b & n_2.c & n_2 \\ n_3.a & n_3.b & d + n_3.c & n_3 \\ -a.d_0 & -b.d_0 & -c.d_0 & -d_1 \end{pmatrix} \qquad \text{------------ (18)}$$

Where d = N.CR' 0 = d0 – d1 = (n1. x0 + n2. Y0 + n3.z0) – (n1.a+n2.b +n3.c)
= n1. (x0 – a) + n2. (y0 – b) + n3. (z0 – c)

And d1 = n1.a + n2.b + n3.c

Example 11: Obtain the perspective transformation onto z = – 2 Plane, where the center of projection is at (0, 0, 18).

Solution: Here centre of projection, C (a, b, c) = (0, 0, 18)

$\therefore$ $(n_1, n_2, n_3) = (0, 0, 1)$

and Reference point R_0 $(x_0, y_0, z_0) = (0, 0, -2)$

$\therefore$ $d_0 = (n_1x_0 + n_2.y_0 + n_3z_0) = -2$

$d_1 = (n_1.a + n_2.b + n_3.\ c) = 18$

we know that the general perspective transformation when cop is not at the origin is given by:

$$\begin{pmatrix} d + n_1.a & n_1.b & n_1.c & n_1 \\ n_2.a & d + n_2.b & n_2.c & n_2 \\ n_3.a & n_3.b & d + n_3.c & n_3 \\ -a.d_0 & -b.d_0 & -c.d_0 & -d_1 \end{pmatrix}$$

$$= \begin{pmatrix} -20 & 0 & 0 & 0 \\ 0 & -20 & 0 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 36 & -18 \end{pmatrix} = \begin{pmatrix} -20 & 0 & 0 & 0 \\ 0 & -20 & 0 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & -2 & 1 \end{pmatrix}$$

Example 12: Find the perspective transformation matrix on to z = 5 plane, when the c.o.p is at origin.

Solution. Since z = 5 is parallel to z = 0 plane, the normal is the same as the unit vecter 'k'.

$\therefore$ $(n_1, n_2, n_3) = (0, 0, 1)$

and the Reference point R0 $(x_0, y_0, z_0) = (0, 0, 5)$

$d_0 = n_1.\ x_0 + n_2.\ y_0 + n_3.\ z_0) = 5$

we know the general perspective transformation, when cop is at origin is given by:

$$\begin{pmatrix} d_0 & 0 & 0 & n_1 \\ 0 & d_0 & 0 & n_2 \\ 0 & 0 & d_0 & n_3 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

☞ **Check Your Progress 4**

1) Determine the vanishing points for the following perspective transformation matrix:

$$\begin{bmatrix} 8.68 & 5.6 & 0 & 2.8 \\ 0 & 20.5 & 0 & 4.5 \\ 7.0 & 800 & 0 & 2.0 \\ 5.3 & 7.3 & 0 & 3.0 \end{bmatrix}$$

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2) Find the three-point perspective transformation with vanishing points at $V_x = 5$, $V_y = 5$ and $V_z = -5$, for a Given eight vertices of a cube A (0, 0, 1), B (1, 0, 1), C (1, 1, 1) D (0, 1, 1), E (0, 0, 0), F (1, 0, 0), G (1, 1, 0), H (0, 1, 0).

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2.3 SUMMARY

- Projection is basically a transformation (mapping) of 3D objects on 2D screen.
- Projection is broadly categorised into Parallel and Perspective projections depending on whether the rays from the object converge at the COP or not.
- If the distance of COP from the projection plane is finite, then we have Perspective projection. This is called perspective because faraway objects look smaller and nearer objects look bigger.
- When the distance of COP from the projection plane is infinite, then rays from the objects become parallel. This type of projection is called parallel projection.
- Parallel projection can be categorised according to the angle that the direction of projection makes with the projection plane.
- If the direction of projection of rays is perpendicular to the projection plane, we have an Orthographic projection, otherwise an Oblique projection.
- Orthographic (perpendicular) projection shows only one face of a given object, i.e., only two dimensions: length and width, whereas Oblique projection shows all the three dimensions, i.e. length, width and height. Thus, an Oblique projection is one way to show all three dimensions of an object in a single view.
- In Oblique projection the line perpendicular to the projection plane are *foreshortened* (Projected line length is shorter then actual line length) by the direction of projection of rays. The direction of projection of rays determines the amount of foreshortening.
- The change in length of the projected line (due to the direction of projection of rays) is measured in terms of foreshortening factor, f, which is defined as the ratio of the projected length to its true length.

- In Oblique projection, if foreshortening factor f=1, then we have cavalier projection and if f=1/2 then cabinet projection.
- The plane of projection may be perpendicular or may not be perpendicular to the principal axes. If the plane of projection is perpendicular to the principal axes then we have *multiview* projection otherwise *axonometric* projection.
- Depending on the foreshortening factors, we have three different types of Axonometric projections: Isometric (all foreshortening factors are equal), Dimetric (any two foreshortening factors equal) and Trimetric (all foreshortening factors unequal).
- In perspective projection, the parallel lines appear to meet at a point i.e., point at infinity. This point is called vanishing point. A practical example is a long straight railroad track, where two parallel railroad tracks appear to meet at infinity.
- A perspective projection can have at most 3 principal vanishing points (points at infinity w.r.t. x, y, and z-axes, respectively) and at least one principle vanishing point.
- A single point perspective transformation with the COP along any of the coordinate axes yields a single vanishing point, where two parallel lines appear to meet at infinity.
- Two point perspective transformations are obtained by the concatenation of any two one-point perspective transformations. So we can have 3 two-point perspective transformations, namely P_{per-xy}, P_{per-yz}, P_{per-xz} .
- Three point perspective transformations can be obtained by the composition of all the three one-point perspective transformations.

2.4 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Consider a following *Figure m*, where a given line AB is projected to A' B' on a projection plane.

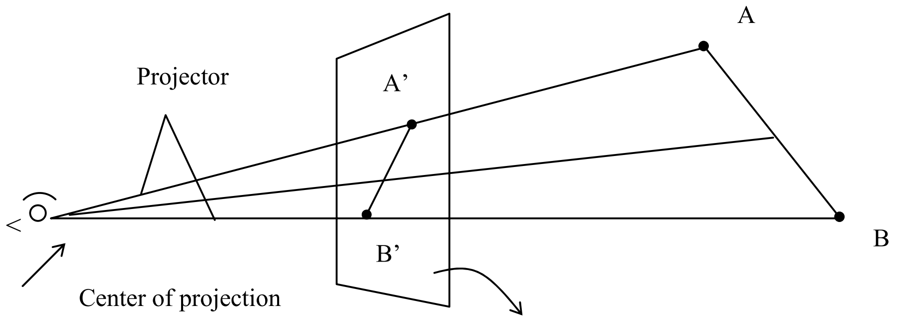


Figure m

a) **Center of projection (cop)**: In case of perspective projection, the rays from an object converge at the finite point, known as center of projection (cop). In *Figure 1*, 0 is the center of projection, where we place our eye to see the projected image on the view plane.

b) **Plane of projection**: Projection is basically a mapping of 3D-object on to 2D-screen. Here 2D-screen, which constitutes the display surface, is known as plane of projection/view plane. That a plane (or display surface), where we are projecting an image of a given 3D-object, is called a plane of projection/view plane. *Figure 1* shows a plane of projection where a given line AB is projected to A'B'.

c) **Projector**: The mapping of 3D-objects on a view plane are formed by projection rays, called the projectors. The intersection of projectors with a view plane form the projected image of a given 3D-object (see *Figure 1*).

d) **Direction of projection**: In case of parallel projection, if the distance of cop from the projection plane is infinity, then all the rays from the object become parallel and will have a direction called "direction of projection". It is denoted by d = (d_1,d_2,d_3), where d_1, d_2 and d_3 make an angle with positive side of *x*, *y* and *z* axes, respectively (see *Figure n*)

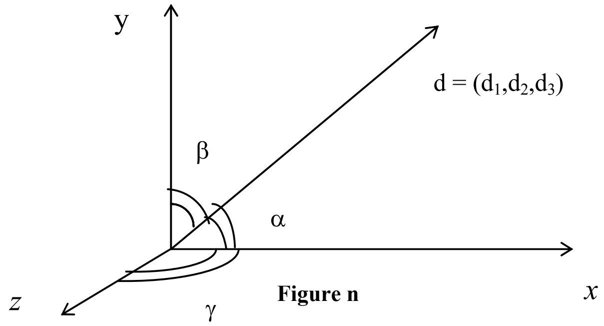


Figure n

The Categorisation of parallel and perspective projection is based on the fact whether coming from the object converge at the cop or not. If the rays coming from the object converges at the centre of projection, then this projection is known as perspective projection, otherwise parallel projection.

Parallel projection can be categorized into orthographic and Oblique projection.

A parallel projection can be categorized according to the angle that the direction of projection $\overline{d}$ makes with the view plane. If $\overline{d}$ is $\perp^r$ to the view plane, then this parallel projection is known as orthographic, otherwise Oblique projection.
Orthographic projection is further subdivided into multiview view plane parallel to the principal axes)

Axonometric projection (view plane not to the principal axes).

Oblique projection is further subdivided into cavalier and canbinet and if f = ½ then cabinet projection.

The *Figure O* shows the Taxonomy of projections:

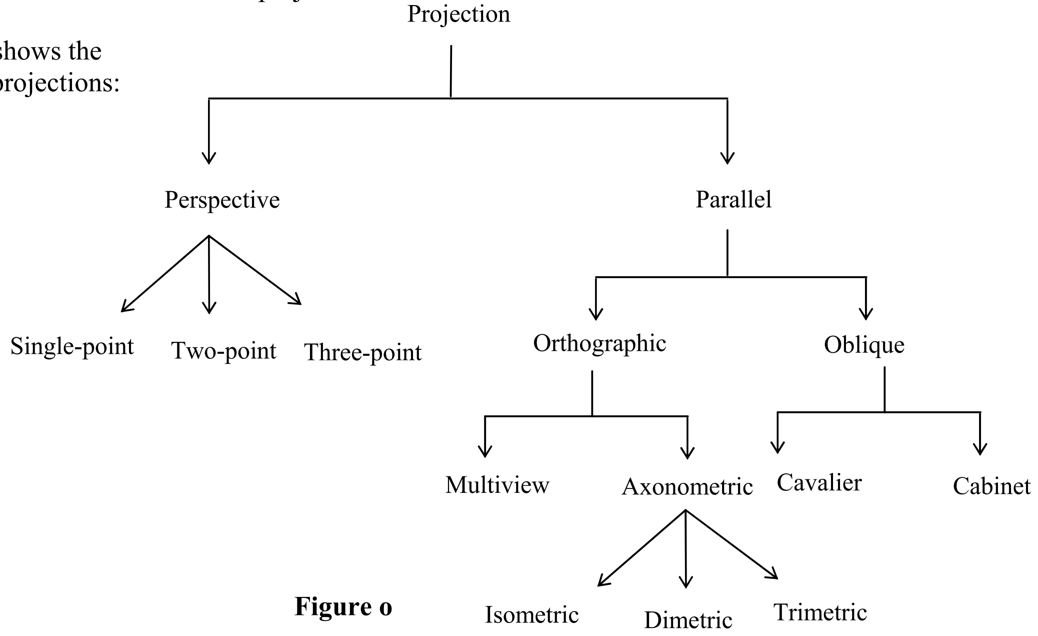


Figure o

3) C

Check Your Progress 2

1) C

2) We know that, the parallel projections can be categorized according to the angle that the direction of projection $\overline{d} = (d_1, d_2, d_3)$ makes with the projection plane. Thus, if direction of projection $\overline{d}$ is $\perp^r$ to the projection plane then we have orthographic projection and if the $\overline{d}$ is not $\perp^r$ to the projection plane then we have oblique projection.

3) The ratio of projected length of a given line to its true length is called the foreshortening factor w.r.t. a given direction.
Let AB is any given line segment
Also assume AB || $\overline{a}$.

Then Under parallel projection, AB is projected to A'B'; The change in the length of projected line is measured in terms of foreshortening factor. f.

$$\therefore \ f = \frac{|A'B'|}{|AB|}$$

Depending on foreshortening factors, we have (3) different types of Axonometric projections:

- Isometric
- Diametric
- Trimetric

When all foreshortening factors along the x-, y- and z-axes are equal, i.e., $f_x = f_y = f_z$, then we have Isometric projection, i.e., the direction of projection makes equal angle with all the positive sides of x, y, and z-axes, respectively.

Similarly, if any two foreshortening factors are equal, i.e., $f_x = f_y$ or $f_y = f_z$ or $f_x = f_z$ then, we have Diametric projection. If all the foreshortening factors are unequal $\overline{d}$ makes unequal angles with *x*, *y*, and *z*-axes/, then we have Trimetric projection.

4) Refer 2. 3. 1. 2 Isometric projection.

5) For orthographic projection, Normal vector N should be parallel to the direction of projection vecter, $\overline{d}$.

i.e. $\overline{d} = k\overline{N}$ where k is a constant.

$(-1, 0, 0) = k(1, 0, -1)$

This is not possible

Hence, the projection plane is not perpendicular to the direction of projection. Hence it is not an orthographic projection.

6) The transformation matrix for cavalier and cabinet projections are given by:

$$P_{cav} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ f.\cos\theta & f.\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \cos 45° & \sin 45° & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{-----1)}$$

$$\&P_{cab}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ f.\cos\theta & f.\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{2}.\sin 30° & \frac{1}{2}.\cos 30° & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.43 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{---(2)}$$

The given pyromid can be shown by the following *Figure p*.

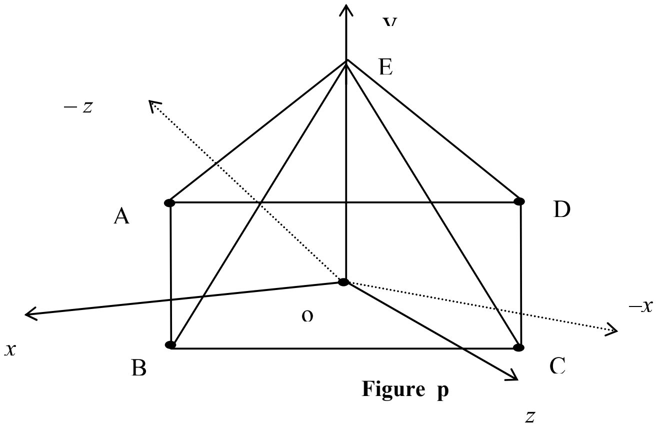


Figure p

The vertices of the pyramid are:

A (2, 0, – 2), B (2, 0, 2), C (– 2, 0, 2)
D (– 2, 0, – 2), E (0, 10, 0)

Using the projection matrices from (1) and (2), we can easily compute the new vertices of the pyramid for cavalier and cabinet projections. (refer Example 4).

Check Your Progress 3

1) Let p (*x, y, z*) be any point in 3D and the cop is E (0, 0, 0).

The parametric equation of the ray, starting from E and passing through p is:

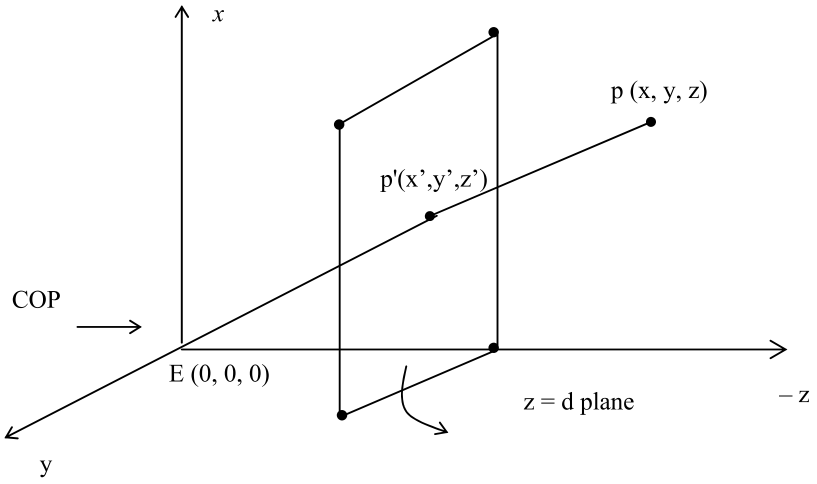


Figure q

E + t (P – E), t > 0
= (0, 0, 0) + t [(*x, y, z*) – (0, 0, 0)]
= (t.x, t. y, t. z)

For this projected point of p (*x, y, z*) will be:

$$t.\ z = d$$

$$= \quad t = \frac{d}{z} \text{ must be true.}$$

Hence the projected point of p (*x, y, z*) will be:

$$P' = (x', y', z') = \left(\frac{d.x}{z}, \frac{d.y}{z}, d\right) \Rightarrow \text{in homogenous Coordinates } \left(\frac{dx}{z}, \frac{dy}{z}, d,1\right)$$

$$= (dx, dy, dz, z)$$

In matrix form:

$$(x', y', z', 1) = (x, y, z, 1)\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

2) Since the cube is first translated by –0.5 units in the *x* and *y*-directions, to get the centred cube on the *z*-axis.

The transformation matrix for translation is:

$$[T_{x,y}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & -0.5 & 0 & 1 \end{bmatrix} \qquad \text{----------(1)}$$

A single-point perspective transformation onto the z = 0 plane is given by:

$$P_{per,z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{----------(2)}$$

It has a center of projection on the z-axis: at $d = -10 \Rightarrow \frac{1}{d} = -0.1$

From equation (2)

$$P_{per,z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The resulting transformation can be obtained as:

$$[T] = [T_{x,y}].\ [P_{per,z}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ -0.5 & -0.5 & 0 & 1 \end{bmatrix}$$

Thus, the projected points of the centred cube V = [ABCDEFGH] will be:

$$[V'] = [V].[T] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.1 \\ -0.5 & -0.5 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 & -0.5 & 0 & 0.9 \\ 0.5 & -0.5 & 0 & 0.9 \\ 0.5 & 0.5 & 0 & 0.9 \\ -0.5 & 0.5 & 0 & 0.9 \\ -0.5 & -0.5 & 0 & 1 \\ 0.5 & -0.5 & 0 & 1 \\ 0.5 & 0.5 & 0 & 1 \\ -0.5 & 0.5 & 0 & 1 \end{bmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{bmatrix} -0.56 & -0.56 & 0 & 1 \\ 0.56 & -0.56 & 0 & 1 \\ 0.56 & 0.56 & 0 & 1 \\ -0.56 & -0.56 & 0 & 1 \\ -0.5 & -0.5 & 0 & 1 \\ 0.5 & -0.5 & 0 & 1 \\ 0.5 & 0.5 & 0 & 1 \\ -0.5 & 0.5 & 0 & 1 \end{bmatrix}$$

3) A unit cube is placed at the origin such that its 3-edges are lying along the x,y, and z-axes. The cube is rotated about the y-axis by 30°. Obtain the perspective projection of the cube viewed from (80, 0, 60) on the z= 0 plane.

3) Rotation of a cube by 30° along y-axis,

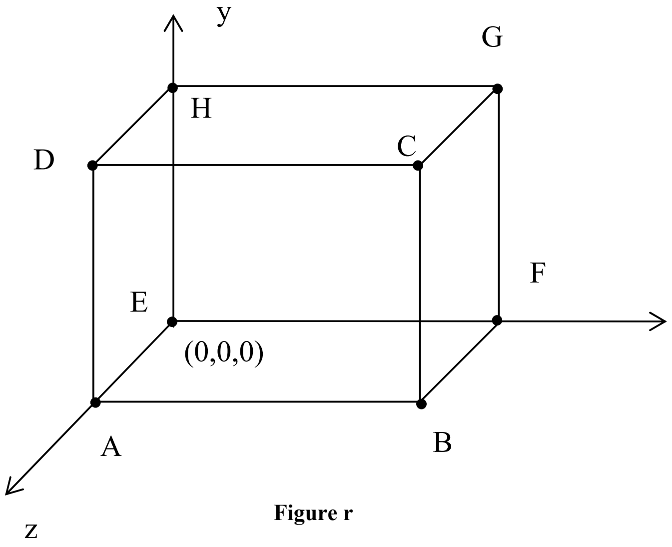


Figure r

$$[R_y]_{30°} = \begin{bmatrix} \cos 30° & 0 & -\sin 30° & 0 \\ 0 & 1 & 0 & 0 \\ \sin 30° & 0 & \cos 30° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/2 & 0 & -1/2 & 0 \\ 0 & 1 & 0 & 0 \\ 1/2 & 0 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.86 & 0 & -0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5 & 0 & 8.6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let p (*x, y, z*) be any point of a cube in a space and p' (x', y', z') is its projected point onto z = 0 plane.

The pametric equation of a line, starting from E (80, 0, 60) and passing through P (x, y, z) is:

$$E + t\,(P - E),\ 0 < t\ \infty.$$
$$= (80, 0, 60) + t\,[(x, y, z) - (80, 0, 60)]$$
$$= (80, 0, 60) + t[\,(x - 80), y, (z - 60)]$$
$$= [t\,(x - 80) + 80,\ t.y,\ t.\,(z - 60) + 60]$$

Assume point P' can be obtained, when t = t*

$\Rightarrow$ P' = (x', y', z') = [t* (x – 80) + 80, t*.y, t* (z – 60) + 60]

Since point p' lies on z = 0 plane, so

$$t^*\,(z - 60) + 60 = 0 \Rightarrow t^* = \frac{-60}{z - 60}$$

$$\Rightarrow p' = (x', y', z') = \left(\frac{-60.x + 80.z}{z - 60}, \frac{-60.y}{z - 60}, 0\right)$$

In Homogeneous coordinates system:

$$P'\,(x', y', z', 1) = \left(\frac{-60.x + 80.z}{z - 60}, \frac{-60.y}{z - 60}, 0, 1\right)$$
$$= (-60.\,x + 80.\,z, -60.\,Y, 0, z - 60)$$

In Matrix form:

$$(x', y', z', 1) = (x, y, z, 1). \begin{bmatrix} -60 & 0 & 0 & 0 \\ 0 & -60 & 0 & 0 \\ 80 & 0 & 0 & 1 \\ 0 & 0 & 0 & -60 \end{bmatrix} \text{------------(1)}$$

$$P'_n = P_n.\ P_{par,\,z} \qquad \text{-------------(2)}$$

Since a given cube is rotated about y-axis by 30°, so the final projected point p' (of a cube on z = 0 plane) can be obtained as follows:

$$P'_n = P_n.\ [R_y]_{30°}.\ P_{par,\,z}$$

$$(x', y', z', 1) = (x, y, z, 1)\ . \begin{bmatrix} 0.86 & 0 & -0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.86 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} -60 & 0 & 0 & 0 \\ 0 & -60 & 0 & 0 \\ 80 & 0 & 0 & 1 \\ 0 & 0 & 0 & -60 \end{bmatrix}$$

$$(x', y', z', 1) = (x, y, z, 1) \begin{bmatrix} 91.9 & 0 & 0 & -0.5 \\ 0 & -60 & 0 & 0 \\ 38.8 & 0 & 0 & 0.86 \\ 0 & 0 & 0 & -60 \end{bmatrix} \text{-------------(3)}$$

$$P''n = P.\ P_{par},\ z,\ 30°$$

This equation (3) is the required perspective transformation. Which gives a coordinates of a projected point P' (x', y', z') onto the z = 0 plane, when a point P (*x*, y, z) is viewed from E (80, 0, 60).

Thus, all the projected points of a given cube can be obtained as follows:

$$P' = V.\ Ppar,\ z,\ 30° = \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{matrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 90.9 & 0 & 0 & -0.5 \\ 0 & -60 & 0 & 0 \\ 38.8 & 0 & 0 & 0.86 \\ 0 & 0 & 0 & -60 \end{bmatrix}$$

$$= \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{bmatrix} 38.8 & 0 & 0 & -59.14 \\ 129.7 & 0 & 0 & -59.64 \\ 129.7 & -60 & 0 & -59.64 \\ 38.8 & -60 & 0 & -60.86 \\ 0 & 0 & 0 & -60.0 \\ 90.9 & 0 & 0 & -60.5 \\ 90.9 & -60 & 0 & -60.5 \\ 0 & -60 & 0 & -60.0 \end{bmatrix} = \begin{matrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{matrix} \begin{bmatrix} -0.72 & 0 & 0 & 1 \\ -2.17 & 0 & 0 & 1 \\ -2.17 & 1.01 & 0 & 1 \\ -0.64 & 0.99 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ -1.50 & 0 & 0 & 1 \\ -1.50 & 0.99 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Hence, A' = (– 0.72, 0, 0), B' = (– 2.17, 0, 0), C' = (– 2.17, 1.01, 0)
D' = (– 0.64, 0.99, 0), E' = (0, 0, 0), F' = (– 1.5, 0, 0)
G' = (– 1.50, 0.99, 0) and H' = (0, 1, 0).

Check Your Progress 4

1) The given perspective transformation matrix can be written as:

From Rows one, two and three from equation matrix (I), the vanishing point w.r.t. x, y and z axis, will be:

$C_x = (3.1, 2.0, 0)$
$C_y = (0, 4.56, 0)$
$C_z = (3.5, 4.0, 0)$

2) From the given V.P., we can obtain the corresponding center of projections. Since vanishing points: $V_x = 5$, $V_y = 5$ and $V_z = -5$, hence center of projections is at:

$C_x = -5$, $C_y = -5$ and $C_z = 5$

$$\therefore 1/d_1 = \frac{1}{5} = 0.2,\ \frac{1}{d_2} = \frac{1}{5} = 0.2 \text{ and } \frac{1}{d_3} = \frac{-1}{5} = -0.2$$

Hence, the 3 – point perspective transformation is:

$$P_{per-xyz} = \begin{bmatrix} 1 & 0 & 0 & 0.2 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 1 & -0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{------------(I)}$$

Thus by multiplying v = [ABCDEFGH] with projection matrix (I), we can obtain the transformed vertices of a given cube.

UNIT 1 CURVES AND SURFACES

Structure **Page Nos.**

1.1	Introduction	5
1.2	Objectives	6
1.3	Polygon Representation Methods	6
	1.3.1 Polygon Surfaces	7
	1.3.2 Polygon Tables	7
	1.3.3 Plane Equation	10
	1.3.4 Polygon Meshes	14
1.4	Bezier Curves and Surfaces	15
	1.4.1 Bezier Curves	16
	1.4.2 Properties of Bezier Curves	20
	1.4.3 Bezier Surfaces	25
1.5	Surface of Revolution	27
1.6	Summary	31
1.7	Solution and Answers	31

1.1 INTRODUCTION

In CS-60, Block 2 and 4 we have studied the technique of drawing curves in different coordinate systems. Also we got the idea that it is the revolution of a curve about some axis that gives rise to an object enclosing some volume and area. For better understanding, just think how a potter works to create vessels of different shapes. He just put a lump of wet soil on the disc which is revolving about its axis at high speed, then his/her fingers and palm works as a curve, in contact with the wet soil. Thus, it is the curve which revolves the some axis to produce vessels of the shape s/he desires. In this unit let us study some of the practical implementations of the concepts studied in CS-60 to computer graphics. This will help a lot because in nature God has created everything with a level of fineness, that if human beings try to achieve that level in their created art (whether computer generated or not) such that it is quite close to reality, then one has to excessively make use of curves and surfaces in a balanced format, and the balance is provided by mathematics. So with the edge of mathematics computer graphics we can achieve realism. In this unit, we will study the polygon representation methods – these methods are quite important because it's the polygon that constitutes every closed object like tree, clouds, ball, car, etc., to be represented through graphics. Further, each polygon has its own mathematical equation which works as the generating function for that polygon. Under this topic we will discuss polygon tables, polygon meshes and equation of plane. A study of these topics will provide you a computer oriented approach to understand the implementation of mathematical concepts. We are going to discuss one more important topic in this unit, which is Bezier Curves and their properties. It's the Bezier curves which have revolutionised the field of computer graphics and opened a new arena, i.e., automobile sector, for analysis and designing of automobiles. Inspired with this achievement, scientists have worked hard and now there is no area which is complete without computer graphics and animation. In this unit, we will deal with fitting curves to the digitized data. Two techniques are available for obtaining such curves *cubic spline and parabolicaly blended curves.* These are based on curve fitting techniques. That is, they are not an approximate method but fit the curve point exactly. We will also discuss curve fairing techniques like Bezier and B spline curves, used to approximate the curve when you are not having a proper knowledge of the shape of the curve. Last but not the least, we will discuss the concept of surface of revolution. It is an important topic because the products which are designed are in fact the surfaces enclosed by the revolution of the curve about some axis. So let us begin our journey.

1.2 OBJECTIVES

After going through the unit, you should be able to:

- implement the methods used to represent a polygon;
- deduce equation of plane and explain the need of planes in graphics;
- discuss various curves and surface representation schemes;
- discuss the piecewise cubic polynomial equation and their need in object representation;
- describe Bezier curve/surface and their properties, and
- discuss the concept of surface of revolution.

1.3 POLYGON REPRESENTATION METHODS

Any scene to be created through computer graphics may contain a variety of objects, some of them natural and some manmade. Thus, to achieve realism in our scene we are not having any specific single method which handles the realistic representational complexities of all components (whether natural or manmade) in a scene.

So let us have an idea of the basic representational schemes available:

- Polygon and quadric surfaces provide precise description for simple Euclidean objects like polyhedrons, ellipsoids.
- Spline surfaces and construction techniques are useful for designing aircraft wings, gears, and other engineering structures with curved surfaces
- Procedural methods such as fractals constructions and particle systems give us accurate representations for the natural objects like clouds, trees etc.
- Physically based modeling methods using systems for interacting forces can be used to describe the non-rigid behaviors of piece of jelly, or a piece of cloth.
- Octrees encoding are used to represent internal features of the objects, such as those obtained from medical CT images, volume renderings and other visualization techniques.

The representational schemes of solid objects are divided into two broad categories:

- **Boundary representations:** Here the 3D object is represented as a set of surfaces that separate the object interior from the environment. Examples are polygonal facets and spline patches. For better understanding consider *Figure 1.*

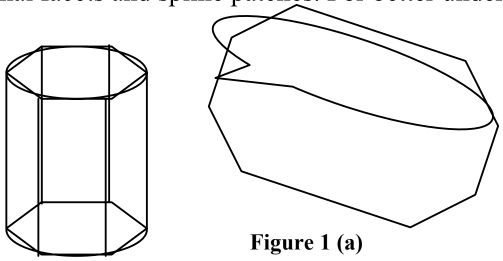


Figure 1 (a)

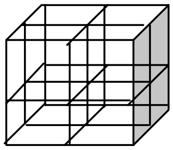


Figure 1 (b)

- **Space partitioning representations:** These are used to describe the interior properties, by partitioning the spatial regions containing an object into a set of small non-overlapping contiguous solids (usually cubes). Example: Octree (which is the space partitioning description of 3D object). For a better understanding consider *Figure 2.*

.

Out of the various representational techniques mentioned above, the most commonly used boundary representation mechanism for representing 3D objects is,

using a set of polygon surfaces to enclose the object interior. Let us discuss this mechanism in our next section.

1.3.1 Polygon Surfaces

From *Figure 1* and *Figure 2* it is quite clear that it is possible to store object's description as a set of surface polygons and the same is actually done by many graphic systems, actually this way of object description fastens the rendering and display of object surfaces. The approach is beneficial because all the surfaces can now be described with linear equations and hence polygonal description of the surfaces is referred as "standard graphic objects". Very many times it is the polygonal representation which is available to describe the object but there are other schemes like spline surfaces which are converted to polygonal representation for processing.

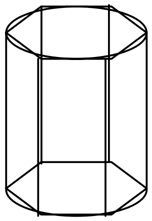


Figure 2

Consider *Figure 3* where the cylindrical surface is represented as a mesh of polygons. The representation is known as wire frame representation which can quickly describe the surface structure. On the basis of the structure a realistic rendering can be performed by interpolating the shading patterns across the polygon surfaces. Thus polygon mesh representation of the curved surface is actually dividing a curved surface into polygon facets. This improves and simplifies the process of rendering (transforming a 3D scene to 2D scene with least loss of information like height, depth etc). Now the objects are composed of standard graphic objects (polygon surfaces). Each graphic object needs some method for its description which could be a polygon table or equation etc. So, let us study the procedures to represent a polygon surface.

1.3.2 Polygon Tables

Every polygon is analogous to a graph G(V,E). We have studied graphs and their theory in detail in MCS-033. Keeping in mind the analogy we can say that a polygon surface can be specified with as a set of vertex coordinates and associated attribute parameters (the attributes may be colour, contrast, shading, etc). Most systems use tables to store the information entered for each polygon, and it's the data in these tables which is then used for subsequent processing, display and manipulation of the objects in the scene. As the data required to be stored for an object in the scene involves both the geometric information and attributes associated with the object, so polygon data tables can be organised into two groups:

- **Attribute tables:** This table holds object information like transparency, surface reflexivity, texture characteristics, etc., of an object in the scene.

- **Geometric tables:** This table stores the information of vertex coordinates and parameters like slope for each edge, etc. To identify the spatial orientation of polygon surface.

In order to store geometric information of a polygon properly, this table is further bifurcated into three more tables:

a) **Vertex table:** Holds coordinate values of vertices in the object.

b) **Edge table:** Holds pointers back in to the vertex table for identification of the vertices related to each polygon edge.

c) **Polygon table or polygon surface table:** Holds pointers back into the edge table for identification of the edges related to the polygon surface under construction.

This tabular representation of a polygon surface is shown in *Figure 4*. Such representations helps one to quickly refer to the data related to a polygon surface. Also, when the data is put for processing then the processing can be quite efficient, leading to efficient display of the object under consideration.

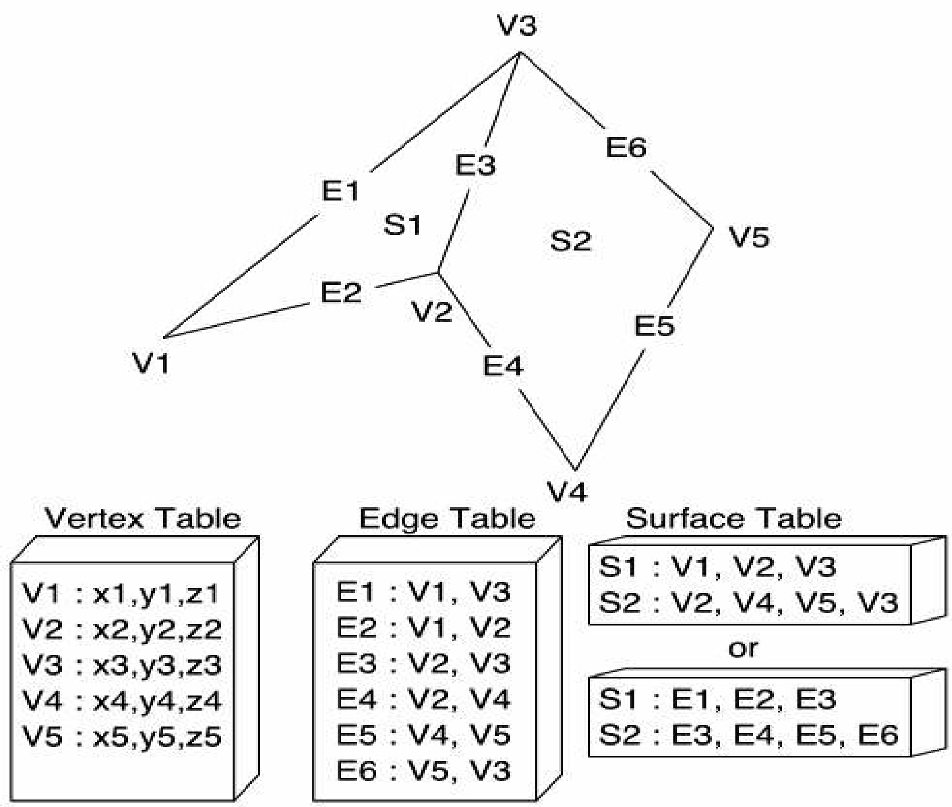


Figure 3

Some basic tests that should be performed before producing a polygon surface by any graphic package:

1) every vertex is listed as an endpoint for at least two edges,
2) every edge is part of at least one polygon,
3) every polygon is closed,
4) each polygon has at least one shared edge,
5) if the edge table contains pointer to polygons, every edge referenced by a polygon pointer to polygon, every edge referenced by a polygon pointer has a reciprocal pointer back to polygon.

Example 1: Set up a geometric data table for an 3d rectangle.

Solution:

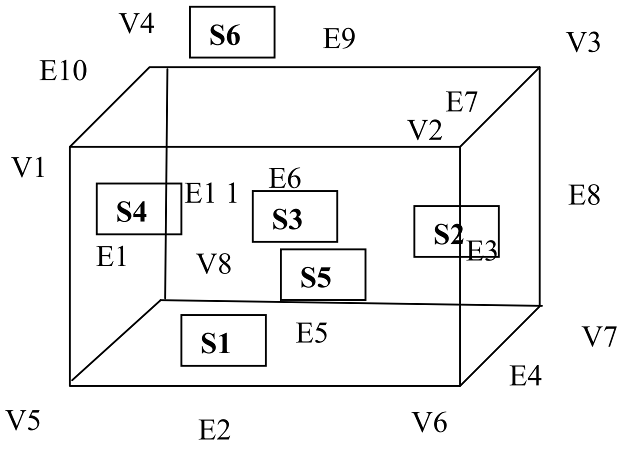


Figure 4

Vertex Table		**Edge Table**		**Polygon Surface Table**	
V1	X1,y1,z1	E1	V1,v5	S1	V1,v2,v6,v5
V2	X2, y2, z2	E2	V5,v6	S2	V2,v6,v7,v3
V3	X3, y3, z3	E3	V6,v2	S3	V8,v7,v3,v4
V4	X4, y4, z4	E4	V6,v7	S4	V1,v4,v5,v8
V5	X5, y5, z5	E5	V7,v8	S5	V8,v5,v6,v7
V6	X6, y6, z6	E6	V1,v2	S6	V4,v1,v2,v3
V7	X7,y7,z7	E7	V2,v3		
V8	X8,y8,z8	E8	V7,v3		
		E9	V3,v4		
		E10	V4,v1		
		E11	V1,v2		

☞Check Your Progress 1

1) What do you think about the utility of polygon surface table? Can't we do the implementation of a polygon surface with just a vertex table and an edge table?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) What happens if we expand the edge table such that it also stores the forward pointers into the polygon table?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) Can we extend the vertex table? Give reasons in support of your answer.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

4) What do you think expanding the tables will make error detection easy or difficult?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

5) Set up a geometric data table for a 3d rectangle using only vertex and polygon tables.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

1.3.3 Plane Equation

Plane is a polygonal surface, which bisects its environment into two halves. One is referred to as forward and the other as backward half of any plane. Now the question is, which half is forward and which backward, because both are relative terms. So to remove this dilemma, we use the mathematical representation of planes, i.e., concepts like equations of planes, normal to a plane, etc., which we have already studied in CS-60. Now we have understood that both forward and backward halves are relative terms but w.r.t what? Yes, it's the plane itself in respect of which we can say any point in the surrounding environment is in front or back of the plane. So we consider any point on the plane should satisfy the equation of a plane to be zero,
i.e., $Ax + By + Cz + D=0$. This equation means any point (x,y,z) will only lie on the plane if it satisfies the equation to be zero any point (x,y,z) will lie on the front of the plane if it satisfies the equation to be greater than zero, and any point (x,y,z) will lie on the back of the plane if it satisfies the equation to be less than zero.

Where (x, y, z) is any point on the plane, and the coefficients A, B, C and D are constants describing the spatial properties of the plane? This concept of space partitioning is used frequently in method of BSP (Binary Space Partitioning) trees generation a polygon representation scheme, quite similar to Octrees.

The importance of plane equations is that they help in producing display of any 3 D object, But for that we need to process the input data representation for the object through several procedures, which may include the following steps of processing.

- To transform the modeling and world-coordinate descriptions to viewing coordinates,
- To devise coordinates,
- To identify visible surfaces,
- To apply surface-rendering procedures.

For some of these processes, we need information about the spatial orientation of the individual surface components of the object. This information is obtained from the vertex coordinates values and the equations that describe the polygon planes. Let us

study how we can determine the equation of any plane. The equation of a plane, say $ax + by + cz + d = 0$, can be determined (generally) in 2 ways:

(1) Say we are given a point $P_0 (x_0, y_0, z_0)$ which lies on the plane and say $\overline{N_i}$ be the . normal to that plane, but we are not given the equation of plane. Then the straight forward procedure to find the equation of plane is:

- choose any other point P (x, y, z) on plane
- determine the line joining point P and P_0 i.e., $\overline{P_0P} = (x - x_0, y - y_0, z - z_0)$
- take dot product of the line $\overline{P_0P}$ and normal to the plane i.e., $\overline{N}$

  As Normal is perpendicular to any line on the plane so the result of the dot product should be zero. Therefore,

$$\begin{aligned}\overline{P_0P}\,.\,\overline{N} &= (x - x_0, y - y_0, z - z_0).\,(n_1, n_2, n_3) = 0\\ &= (x - x_0)\, n_1 + (y - y_0)\, n_2 + (z - z_0)\, n_3 = 0\\ &= n_1x + n_2y + n_3z = (n_1x_0 + n_2y_0 + n_3z_0)\end{aligned}$$

$\therefore$ equation of plane is: $n_1x + n_2y + n_3z - (n_1x_0 + n_2y_0 + n_3z_0) = 0$.

(2) In the equation $Ax + By + Cz + D = 0$ for a plane surface ,where (x, y, z) is any point on the plane, and the coefficients A, B, C and D are constants describing the spatial properties of the plane. If the values of A,B,C, and D are not given then we can obtain the values of A, B, C and D by solving a set of three plane equations using the coordinate values for three non collinear points in the plane, which are say(x_1,y_1,z_1) , (x_2,y_2,z_2) , (x_3,y_3,z_3)

For this purpose, we can select the following set of simultaneous linear plane equations for the ratios A/D, B/D, and C/D:

$$\left.\begin{aligned}&Ax_1 + By_1 + Cz_1 + D = 0\\ &Ax_2 + By_2 + Cz_2 + D = 0\\ &Ax_3 + By_3 + Cz_3 + D = 0\\ \text{Then}\quad &(A/D)x_k + (B/D)y_k + (C/D)z_k = -1, \qquad k = 1, 2, 3\end{aligned}\right\} \quad (1)$$

The solution for this set of equations can be obtained in determinant form, using Cramer's rule, as

$$\left.\begin{aligned}A &= \begin{vmatrix}1 & y_1 & z_1\\ 1 & y_2 & z_2\\ 1 & y_3 & z_3\end{vmatrix} \qquad B = \begin{vmatrix}x_1 & 1 & z_1\\ x_2 & 1 & z_2\\ x_3 & 1 & z_3\end{vmatrix}\\ C &= \begin{vmatrix}x_1 & y_1 & 1\\ x_2 & y_2 & 1\\ x_3 & y_3 & 1\end{vmatrix} \qquad D = -1\begin{vmatrix}x_1 & y_1 & z_1\\ x_2 & y_2 & z_2\\ x_3 & y_3 & z_3\end{vmatrix}\end{aligned}\right\} \quad (2)$$

Expanding the determinants, we can write the calculations for the plane coefficients in the form

$$\left.\begin{aligned}A &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)\\ B &= z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)\\ C &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)\\ D &= -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)\end{aligned}\right\} \quad (3)$$

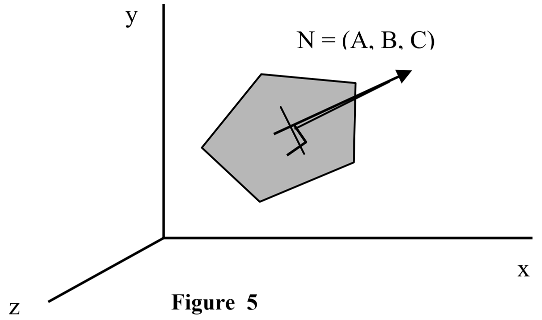


Figure 5

Note: Orientation of a plane surface in space can be described with the normal vector to the plane, as shown in the *Figure 5*. Further the Cartesian component of vector N, normal to the surface of the plane described by equation $Ax + By + Cz + D = 0$, is given by (A,B,C) where parameters A, B, and C are the plane coefficients calculated in equations above.

While dealing with polygon surfaces we have understood that polygon tables play a vital role in storing the information about the polygon. So, the vertex values and other information are entered into the polygon data structure and hence the values for A, B, C and D are computed for each polygon and stored with the other polygon data.

Since, we are usually dealing with polygon surfaces that enclose an object interior, we need to distinguish between the two sides of the surface. The side of or outward side is the "outside" face. If polygon vertices are specified in a counterclockwise direction when viewing the outer side of the plane in a right-handed coordinate system, the direction of the normal vector will be from inside to outside. This is demonstrated for one plane of a unit cube shown in the *Figure 6*.

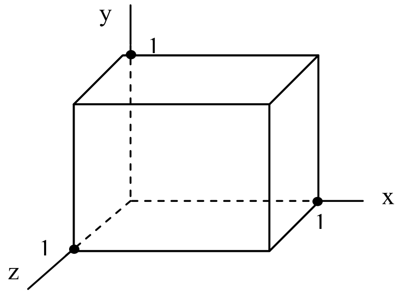


Figure 6

To determine the components of the normal vector for the shaded surface shown in the *Figure 6* of a cube., we select three of the four vertices along the boundary of the polygon. These points are selected in a counterclockwise direction as we view from outside the cube toward the origin. Coordinates for these vertices, in the order selected, can be used in Equations. (3) to obtain the plane coefficients:
$A = 1, B = 0, C = 0, D = -1$. Thus, the normal vector for this plane is in the direction of the positive *x*- axis.

The elements of the plane normal can also be obtained using a vector cross product calculations. We again select three vertex positions, V_1, V_2, and V_3, taken in counterclockwise order when viewing the surface from outside to inside in a right-handed Cartesian system. Forming two vectors, one from V_1 to V_2 and the other from V_1 to V_3, we calculate N as the vector cross product:

$$N = (V_2 - V_1) X (V_3 - V_1)$$

This generates values for the plane parameters A, B and C. We can then obtain the value for parameter D by substituting these values and the coordinates for one of the polygon vertices in plane equation and solving for D. The plane equation can be expressed in vector form using the normal N and the position P of any point in the plane as:

$$N \cdot P = -D$$

Plane equations are used also to identify the position of spatial points relative to the plane surfaces of an object. For any point (x, y, z) not on a plane with parameters A, B, C, D, we have:

$$Ax + By + Cz + D \neq 0$$

We can identify the point as either inside or outside the plane surface according to the sign (negative or positive) of $Ax + By + Cz + D$:

if $Ax + By + Cz + D < 0$, the point (x, y, z) is inside the surface
if $Ax + By + Cz + D > 0$, the point (x, y, z) is outside the surface

These inequality tests are valid in a right-handed Cartesian system, provided the plane parameter A, B, C and D were calculated using vertices selected in a counterclockwise order when viewing the surface in an outside-to-inside direction. For example, in the above figure of the cube any point outside the shaded plane satisfies the inequality $x - 1 > 0$, while any point inside the plane has an x-coordinates value less than 1.

Example 2: Find equation of plane which passes through point P (0, 0, 0) and say the normal to the plane is given by $\overline{N}(1, 0, -1)$?

Solution: Let us use method 1 discussed above to determine the equation of the plane

Given $\overline{N}$ (1, 0, – 1) and P (0, 0, 0) ; equation of plane = ?

say P'(x, y, z) be another point on the plane then line $\overrightarrow{PP'} = (x\text{-}0, y\text{-}0, z\text{-}0) = x\hat{i} + y\hat{j} + z$
k

now determine the dot product of PP' and normal N

$$\overline{PP'} \cdot \overline{N} = 0 \;\Rightarrow n_1x, + n_2y + n_3z - (x_0n_1 + y_0n_2 + z_0n_3) = 0$$
$$1.x + 0.\,y + (-1).z - (0 + 0 + 0) = 0$$
$$x - z = 0 \;\rightarrow \text{plane equation}$$

$\Rightarrow x = z$ is the required plane shown in *Figure 7* below

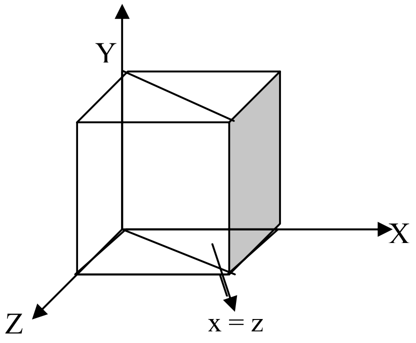


Figure 7

1.3.4 Polygon Meshes

A polygonal surface to be drawn may not be simple and may have enormous curls and curves. Example, a crushed piece of paper, or crushed piece of aluminum foil, etc. In such cases each section of a polygonal surface can be generated (in computer graphics or can be simply drawn) with the help of various standard graphic objects like rectangles, triangles, circles (semicircles), spheres (hemispheres) etc., drawn in a manner that their pattern combination matches with the polygonal surface under construction. This cumulative combination of all standard graphic objects is in fact the mesh or polygonal mesh used to approximate the actual geometry of any complicated object under construction, with the help of the standard graphic objects.

After studying the section 1.3.2 polygon tables, we came to the conclusion that a polygonal surface can be represented with the set of vertices, set of edges and set of surfaces ; which are the general terminologies of nothing but graphs. So we will use this concept here too because, the polygons we need to represent can be arbitrarily large. Thus, it is generally convenient and more appropriate to use a polygon mesh rather than a single mammoth polygon (i.e., single standard graphic object). For example, you can simplify the process of rendering polygons by breaking all polygons into triangles. Triangle renderers can also be implemented in hardware, making it advantageous to break the world down into triangles. Consider below *Figure 8:*

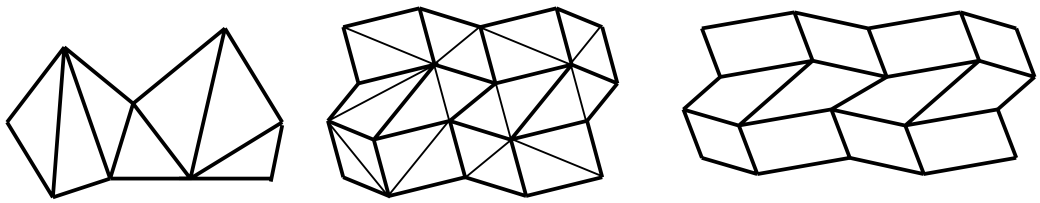


Figure 8 (a) **Figure 8 (b)** **Figure 8 (c)**

Another example where smaller polygons are better is the Inventor lighting model. Inventor computes lighting at vertices and interpolates the values in the interiors of the polygons. By breaking larger surfaces into meshes of smaller polygons, the lighting approximation is improved. From the shown *Figure 8* two important observations are:

- Triangle mesh produces n-2 triangles from a polygon of n vertices.
- Quadrilateral mesh produces (n-1) by (m-1) quadrilaterals from an n x m array of vertices.

It is important to note that specifying polygons with more than three vertices could result in sets of points, which are not co-planar, the reason behind may be the numerical errors or error in selecting the coordinate position of the vertices. Handling non-coplanar vertices is quite difficult, so two ways to handle such situation are:

- Break the polygon into triangles, and deal.
- Approximate *A, B,* and *C* in the plane equation. This can be done either by averaging or by projecting the polygon onto the coordinate planes. *A* should be proportional to the projection in the *yz*-plane, *B* proportional to *xz*, and *C* proportional to xy. High quality graphics system typically model objects with polygon meshes and set up a database of geometric and attribute information to facilitate processing of the polygon facets. Fast hardware implemented polygon renderers are incorporated into such systems with the capability for displaying hundreds of thousands to one millon or more shaded polygon per second including the application of surface texture.

☞**Check Your Progress 2**

1) Find equation of plane, which passes through point P (1, 1, 1) and say the normal to the plane is given by $\overline{N}$ (-1, 0, -1).

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) Calculate the equation of plane for the quadrilateral planar polygon described by the four vertices v1 (1,0,1), v2 (1,1,0), v3(0,1,1) and v4 (1,1,1).

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

1.4 BEZIER CURVES AND SURFACES

We had discussed in the previous section of this unit that we can create complex geometries with the help of polygon meshes which are further constituted of standard polygonal objects like triangle, rectangle, square, etc., but apart from this technique to draw complex geometries, we are having some more advanced techniques to do the same job like we can use mathematical equations (parametric equations and polynomial equations), splines, fractals, Bezier curves etc. In this section we will discuss some of these techniques, but to have the flavor of the detailed analysis of these techniques you can refer to books given in suggested readings. Before going on the tour of Bezier curves let us have a brief discussion on other techniques, the technique of using mathematical equations (parametric equations and polynomial equations) to realize the complex natural scenes is not always successful because it requires an enormous number of calculations which consumes an immense amount of processing time. The better technique to generate complex natural scenes is to use fractals. Fractals are geometry methods which use procedures and not mathematical equations to model objects like mountains, waves in sea, etc. There are various kinds of fractals like self-similar, self-affined, etc. This topic is quite interesting but is out of the scope of this unit. Now, let us discuss Bezier curves, which is a Spline approximation method developed by the French engineer Pierre Bezier for use in the design of Renault automobile bodies. Bezier splines have a number of properties that make them highly useful and convenient for curve and surface design. They are also easy to implement. For these reasons, Bezier splines are widely available in various CAD systems, in general graphics packages (such as GL on Silicon Graphics systems), and in assorted drawing and painting packages (such as Aldus Super Paint and Cricket Draw).

Before going into other details of the Bezier curves, we should learn something about the concept of Spline and their representation. Actually Spline is a flexible strip used to produce a smooth curve through a designated set of points known as Control points (it is the style of fitting of the curve between these two points which gives rise to Interpolation and Approximation Splines).We can mathematically describe such a curve with a piecewise cubic Polynomial function whose first and second derivatives are continuous across the various curve sections. In computer graphics, the term spline curve now refers to any composite curve formed with polynomial sections satisfying specified continuity conditions (Parametric continuity and Geometric continuity conditions) at the boundary of the pieces, without fulfilling these conditions no two curves can be joined smoothly. A spline surface can be described with two sets of

orthogonal spline curves. There are several different kinds of spline specifications that are used in graphics applications. Each individual specification simply refers to a particular type of polynomial with certain specified boundary conditions. Splines are used in graphics applications to design curve and surface shapes, to digitize drawings for computer storage, and to specify animation paths for the objects or the camera in a scene. Typical CAD applications for splines include the design of automobile bodies, aircraft and spacecraft surfaces, and ship hulls.

We have mentioned above that it is the style of fitting of the curve between two control points which gives rise to Interpolation and Approximation Splines, i.e., we can specify a spline curve by giving a set of coordinate positions, called control points, which indicates the general shape of the curve. These control points are then fitted with piecewise continuous parametric polynomial functions in one of two ways. When polynomial sections are fitted so that the curve passes through each control point, the resulting curve is said to interpolate the set of control points. On the other hand, when the polynomials are fitted to the general control-point without necessarily passing through any control point, the resulting curve is said to approximate the set of control points. Interpolation curves are commonly used to digitize drawings or to specify animation paths. Approximation curves are primarily used as design tools to structure object surfaces.

A spline curve is defined, modified, and manipulated with operations on the control points. By interactively selecting spatial positions for the control points, a designer can set up an initial curve. After the polynomial fit is displayed for a given set of control points, the designer can then reposition some or all of the control points to restructure the shape of the curve. In addition, the curve can be translated, rotated, or scaled with transformations applied to the control points. CAD packages can also insert extra control points to aid a designer in adjusting the curve shapes.

1.4.1 Bezier Curves

Bezier curves are used in computer graphics to produce curves which appear reasonably smooth at all scales. This spline **approximation method** was developed by French engineer Pierre Bezier for automobile body design. Bezier spline was designed in such a manner that they are very useful and convenient for curve and surface design, and are easy to implement Curves are trajectories of moving points. We will specify them as functions assigning a location of that moving point (in 2D or 3D) to a parameter t, i.e., parametric curves.

Curves are useful in geometric modeling and they should have a shape which has a clear and intuitive relation to the path of the sequence of control points. One family of curves satisfying this requirement are Bezier curve.

The Bezier curve require only two end points and other points that control the endpoint tangent vector.

Bezier curve is defined by a sequence of N + 1 control points, $P_0, P_1, \ldots, P_n$. We defined the Bezier curve using the algorithm (invented by ***DeCasteljeau***), based on recursive splitting of the intervals joining the consecutive control points.

A purely geometric construction for Bezier splines which does not rely on any polynomial formulation, and is extremely easy to understand. The DeCasteljeau method is an algorithm which performs repeated bi-linear interpolation to compute splines of any order.

De Casteljeau algorithm: The control points P0, P1, P2 and P3are joined with line segments called 'control polygon', even though they are not really a polygon but rather a polygonal curve.

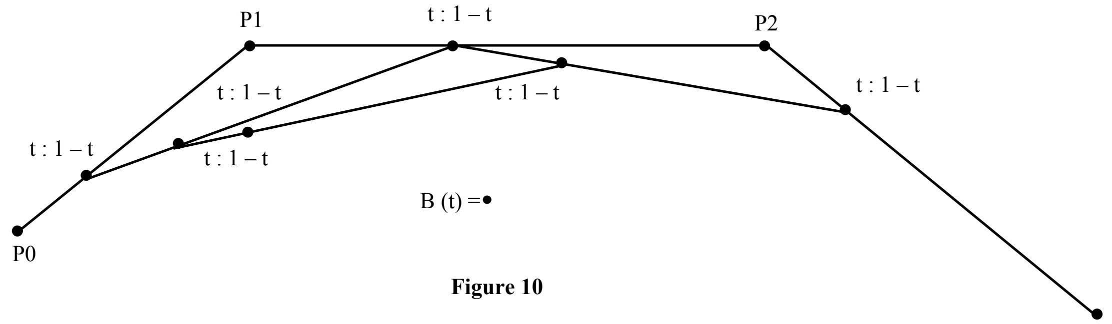


Figure 10

Each of them is then divided in the same ratio t : 1- t, giving rise to the another points. Again, each consecutive two are joined with line segments, which are subdivided and so on, until only one point is left. This is the location of our moving point at time t. The trajectory of that point for times between 0 and 1 is the Bezier curve.

A simple method for constructing a smooth curve that followed a control polygon p with *m-1* vertices for small value of m, the Bezier techniques work well. However, as *m* grows large (*m*>20) Bezier curves exhibit some undesirable properties.

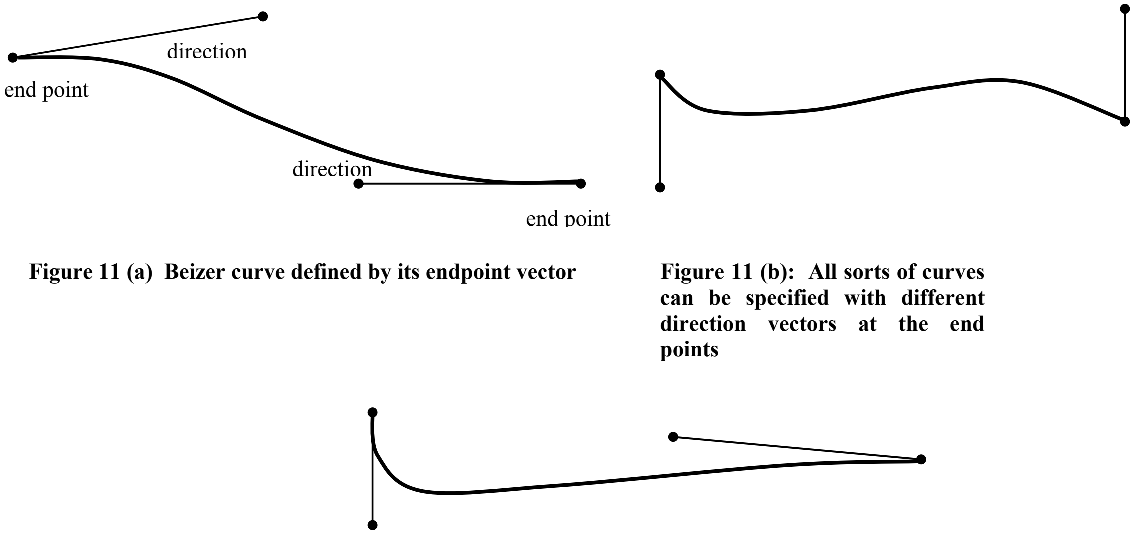


Figure 11 (a) Beizer curve defined by its endpoint vector

Figure 11 (b): All sorts of curves can be specified with different direction vectors at the end points

Figure 11 (c): Reflex curves appear when you set the vectors in different directions

In general, a Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative positions determine the degree of the Bezier polynomial. As with the interpolation splines, a Bezier curve can be specified with boundary conditions, with a characterizing matrix, or with blending function. For general Bezier curves, the blending-function specification is the most convenient.

Suppose we are given n + 1 control-point positions: $p_k = (x_k, y_k, z_k)$, with *k* varying from 0 to n. These coordinate points can be blended to produce the following position vector *P(u)*, which describes the path of an approximating Bezier polynomial function between p_0 and p_n.

$$P(u) = \sum_{k=0}^{n} p_k B_{k,n}(u), \quad 0 \le u \le 1 \qquad (1)$$

The Bezier blending functions $B_{k,n}(u)$ are the Bernstein polynomials.

$$B_{k,n}(u) = C(n,k)u^k(1-u)^{n-k} \qquad (2)$$

Where the C(n, k) are the binomial coefficients:

$$C(n,k) = \ {}^nC_k = \frac{n!}{k!(n-k)!} \qquad (3)$$

equivalently, we can define Bezier blending functions with the recursive calculation

$$B_{k,n}(u) = (1-u)B_{k,n-1}(u) + uB_{k-1,n-1}(u), \qquad n > k \ge 1 \qquad (4)$$

with $BEZ_{k,k} = u^k$, and $B_{0,k} = (1-u)^k$. Vector equation (1) represents a set of three parametric equations for the individual curve coordinates:

$$x(u)\sum_{k=0}^{n} x_k B_{k,n}(u)$$
$$y(u)\sum_{k=0}^{n} y_k B_{k,n}(u) \qquad (5)$$
$$z(u)\sum_{k=0}^{n} z_k B_{k,n}(u)$$

As a rule, a Bezier curve is a polynomial of degree one less than the number of control points used: Three points generate a parabola, four points a cubic curve, and so forth. *Figure 12* below demonstrates the appearance of some Bezier curves for various selections of control points in the xy plane (z = 0). With certain control-point placements, however, we obtain degenerate Bezier polynomials. For example, a Bezier curve generated with three collinear control points is a straight-line segment. And a set of control points that are all at the same coordinate position produces a Bezier "curve" that is a single point.

Bezier curves are commonly found in painting and drawing packages, as well as CAD system, since they are easy to implement and they are reasonably powerful in curve design. Efficient methods for determining coordinate position along a Bezier curve can be set up using recursive calculations. For example, successive binomial coefficients can be calculated as shown below through examples of two-dimensional Bezier curves generated from three, four, and five control points. Dashed lines connect the control-point positions.

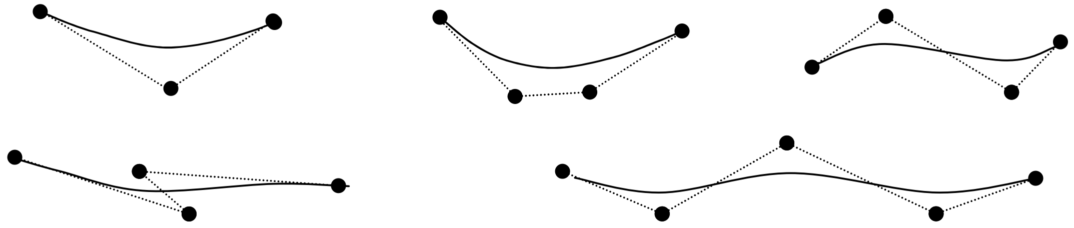


Figure 12

Note:

1) Bezier Curve: $P(u) = \sum_{i=0}^{n} p_i B_{n,i}(u)$ …………… (1)

Where $B_{n,i}(u) = n_{c_i} u^i (1-u)^{n-i}$ ………….. (2)

$$C(n,i) = nC_i = \frac{n!}{i!(n-i)!} \quad 0 \le u \le 1$$

2) Cubic Bezier curve has n = 3:

$\therefore P(u) = \sum_{i=0}^{3} p_i B_{3,i}(u)$ ………………..(3)

$= p_0 B_{3,0}(u) + p_1 B_{3,1}(u) + p_2 B_{3,2}(u) + p_3 B_{3,3}(u)$

Now, lets find $B_{3,0}(u)$, $B_{3,1}(u)$, $B_{3,2}(u)$, $B_{3,3}(u)$ using above equation

$B_{n,i}(u) = n_{c_i} u^i (1-u)^{n-i}$

a) $B_{3,0}(u) = {}^3C_0 \; u^0 \; (1-u)^{3-0}$

$= \frac{3!}{0!(3-0)!} . 1. (1-u)^3 = (1-u)^3$

b) $B_{3,1}(u) = {}^3C_1 u^1 (1-u)^{3-1} = \frac{3!}{1!(3-1)!} u (1-u)^2$

$= 3u (1-u)^2$

c) $B_{3,2}(u) = {}^3C_2 u^2 \; (1-u)^{3-2} = \frac{3!}{2!(3-2)!} u^2 (1-u)$

$= 3u^2 (1-u)$

d) $B_{3,3}(u) = {}^3C_3 u^3 (1-u)^{3-3} = \frac{3!}{3!(3-3)!} u^3$

Using (a), (b) , (c) & (d) in (5) we get

$P(u) = p_0 (1-u)^3 + 3p_1 u (1-u)^2 \; 3p_2 u^2 (1-u) + p_3 u^3$

Example 4: 1 Given p_0 (1, 1): p_1 (2, 3); p_2 (4, 3); p_3 (3, 1) as vertices of Bezier curve determine 3 points on Bezier curve?

Solution: We know Cubic Bezier curve is

$P(u) = \sum_{i=0}^{3} pi B_{3,i}(u)$

$\Rightarrow P(u) = p_0 (1-u)^3 + 3p_1 u (1-u)^2 + 3p_2 u^2 (1-u) + p_3 u^3$

$P(u) = (1, 1) (1-u)^3 + 3 (2, 3) u (1-u)^2 + 3 (4, 3) u^2 (1-u) + (3, 1) u^3.$

we choose different values of u from 0 to 1.

u = 0: $P(0) = (1, 1) (1-0)^3 + 0 + 0 + 0 = (1, 1)$

u = 0.5: $P(0.5) = (1, 1)(1 - 0.5)^3 + 3(2, 3)(0.5) (1 - 0.5)^2 + 3 (4, 3)(0.5)^2 (1 - 0.5) + (3,1) (0.5)^3$

$= (1, 1) (0.5)^3 + (2, 3) (0.375) + (0.375) (4, 3) + (3, 1) (0.125)$

$= (0.125, 0.125) + (0.75, 1.125) + (1.5, 1.125) + (1.125, 0.125)$

$P(0.5) = (3.5, 2.5)$

u = 1 : $P(1) = 0 + 0 + 0 + (3, 1). 1^3$

$= (3, 1)$

Three points on Bezier curve are, P (0) = (1, 1); P (0.5) = (3.5, 2.5) and P (1) = (3, 1).

1.4.2 Properties of Bezier Curves

A very useful property of a Bezier curve is that it always passes through the first and last control points. That is, the boundary conditions at the two ends of the curve are

$$P(0) = p_0$$
$$P(1) = p_n$$

Values of the parametric first derivatives of a Bezier curve at the end points can be calculated from control-point coordinates as

$$P'(0) = -np_0 + np_1$$
$$P'(1) = -np_{n-1} + np_n$$

Thus, the slope at the beginning of the curve is along the line joining the first two control points, and the slope at the end of the curve is along the line joining the last two endpoint. Similarly, the parametric second derivatives of a Bezier curve at the endpoints are calculated as

$$p''(0) = n)n-1)[(p_2 - p_1) - (p_1 - p_0)]$$
$$p''(1) = n)n-1)[(p_{n-2} - p_{n-1}) - (p_{n-1} - p_n)]$$

Another important property of any Bezier curve is that it lies within the convex hull (convex polygon boundary) of the control points. This follows from the properties of Bezier blending functions: They are all positive and their sum is always 1,

$$\sum_{k=0}^{n} B_{k,n}(u) = 1$$

so that any curve position is simply the weighted sum of the control-point positions. The convex-hull property for a Bezier curve ensures that the polynomial will not have erratic oscillations.

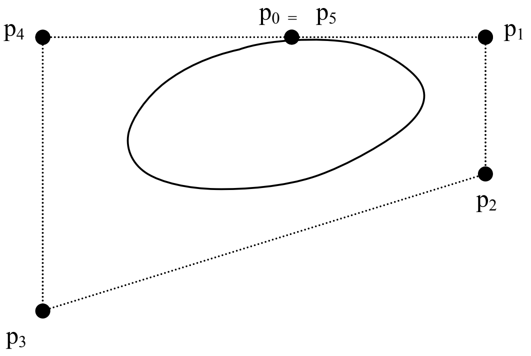


Figure 13 (a): Shows closed Bezier curve generated by specifying the first and last control points at the same location

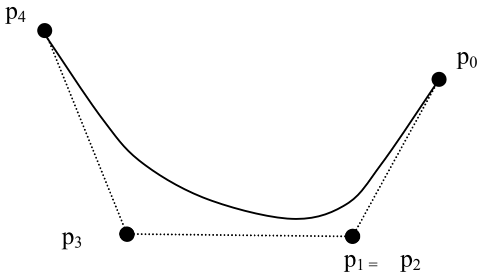


Figure 13 (b): Shows that a Bezier curve can be made to pass closer to a given coordinate position by assigning multiple control points to that position.

Note:

1) Generalising the idea of Bezier curve of degree at n based on n+1 control point p0,…..pn
   $P(0) = P_0$
   $P(1) = p_n$

Values of parametric first derivates of Bezier curve at the end points can be calculated from control point

Coordinates as

$P'(0) = -nP_0 + n P_1$

$P'(1) = -nP_{n-1} + nP_n$

Thus, the slope at the beginning of the curve is along the line joining two control points, and the slope at the end of the curve is along the line joining the last two endpoints.

2) **Convex hull**: For $0 \le t \le 1$, the Bezier curve lies entirely in the convex hull of its control points. The convex hull property for a Bezier curve ensures that polynomial will not have erratic oscillation.

3) Bezier curves are invariant under affine transformations, but they are not invariant under projective transformations.

4) The vector tangent to the Bezier curve at the start (stop) is parallel to the line connecting the first two (last two) control points.

5) Bezier curves exhibit a *symmetry* property: The same Bezier curve shape is obtained if the control points are specified in the opposite order. The only difference will be the parametric direction of the curve. The direction of increasing parameter reverses when the control points are specified in the reverse order.

6) Adjusting the position of a control point changes the shape of the curve in a "predictable manner". Intuitively, the curve "follows" the control point.

There is *no local control* of this shape modification. Every point on the curve (with the exception of the first and last) move whenever any interior control point is moved.

Following examples prove the discussed properties of the Bezier curves

Example 5: To prove: $P(u = 0) = p_0$

Solution: $\because P(u) = \sum_{i=0}^{n} p_i B_{n,i}(u)$

$= p_0 B_{n,0}(u) + p_1 B_{n,1}(u) + \ldots\ldots + p_n B_{n,n}(u) \ldots\ldots\ldots\ldots\ldots (1)$

$B_{n,i}(u) = {}^{n}C_{i}\, u^i (1-u)^{n-i}$

$B_{n,0}(u) = {}^{n}C_{0}\, u^0 (1-u)^{n-0} = \frac{n!}{0!(n-0)!} \cdot 1 \cdot (1-u)^n = (1-u)^n$

$B_{n,1}(u) = {}^{n}C_{1}\, u' (1-u)^{n-1} = \frac{n!}{1!(n-1)!} \cdot u \cdot (1-u)^{n-1}$

We observe that all terms expect $B_{n,0}(u)$ have multiple of u^i ($i = 0$ to n) using these terms with $u = 0$ in (1) we get,

$P(u = 0) = p_0 (1-0)^n + p_1 \cdot 0 \cdot (1-0)^{n-1} \cdot n + 0 + 0 + \ldots\ldots + 0$

$P(u = 0) = p_0$ Proved

Example 6: To prove $\overline{P}(1) = p_n$

Solution: As in the above case we find each term except $B_{n,n}(u)$ will have multiple of $(1-u)^i$ ($i = 0$ to n) so using $u = 1$ will lead to result $= 0$ of all terms except of $B_{n,n}(u)$.

$$B_{n,n}(u) = \frac{n!}{n!\,(n-n)!}\, u^n (1-u)^{n-n} = u^n$$

$$P(u-1) = p_0 \,.\, 0 + p_1 .\, 0 + \ldots\ldots + p_n .\, 1^n$$
$$= p_n$$

Example 7: Prove: $\sum_{i=0}^{n} B_{n,i} = 1$

Solution: By simple arithmetic we know,

$[(1-u)+u]^n = 1^n = 1$ ………………..(1)

expending LHS of (1) binomially we find

$[(1-u)+u]^n = {}^{n}c_0\ (1-u)^n + {}^{n}c_1\ u\,(1-u)^{n-1} + {}^{n}c_2\, u^2\,(1-u)^{n-2} + \ldots.. + {}^{n}c_n$
u^n

$$= \sum_{i=0}^{n} {}^{n}c_i\ u^i (1-u)^{n-i}$$

$[(1-u)+u]^n = \sum_{i=0}^{n} B_{n,i}(u)$ ……………..(2)

by (1) & (2) we get

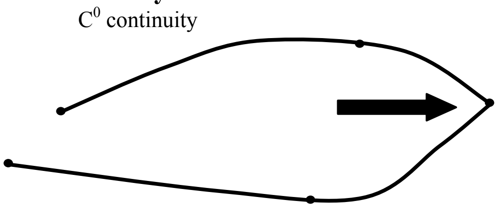
$$\boxed{\sum_{i=0}^{n} B_{n,i}(u) = 1}$$

Note: Proof of following properties of Bezier curves is left as an exercise for the students

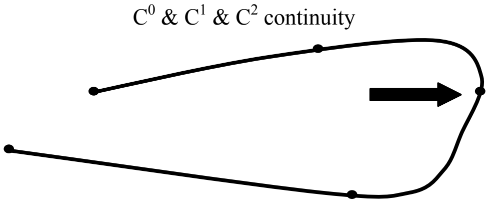
$P'(0) = n\,(p_1 - p_0)$
$P'(1) = n\,(p_n - p_{n-1}) = n\,(p_n - p_{n-1})$
$P''(0) = n\,(n-1)\,(p_0 - 2p_1 + p_2)$
$P''(1) = n\,(n-1)\,(p_n - 2\,p_{n-1} + p_{n-2})$

To ensure the smooth transition from one section of a piecewise parametric curve or any Bezier curve to the next we can impose various continuity conditions at the connection point for ***parametric continuity*** we match parametric derivatives of adjoining curve section at their common boundary.

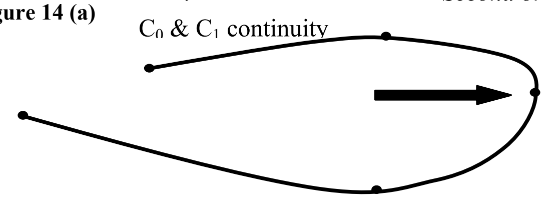
Zero order parametric continuity described by C^0 continuity means curves are only meeting as shown in *Figure 14* while first order parametric continuity referred as C^1 continuity, means that tangent of successive curve sections are equal at their joining point. Second order parametric continuity or C^2 continuity, means that the parametric derivatives of the two curve sections are equal at the intersection. As shown in *Figure 14* below **first order continuity have equal tangent vector but magnitude may not be the same.**



Zero order parametric Continuity
Figure 14 (a)



Second order parametric continuity
Figure 14 (b)



First order parametric continuity
Figure 14 (c)

With the second order continuity, the rate of change of tangent vectors for successive section are equal at intersection thus result in smooth tangent transition from one section to another.

First order continuity is often used in digitized drawing while second order continuity is used in CAD drawings.

Geometric continuity is another method to join two successive curve sections. G^0 continuity is the same as parametric continuity (i.e., two curves sections to be joined must have same coordinate position at the boundary point) i.e., curve section are joined together such that they have same coordinates position at the boundary point. First order geometric continuity $G^{(1)}$ means that the tangent vectors are the same at join point of two successive curves i.e., the parametric first derivative are proportional at the intersection of two successive sections while second order geometric continuity is G^2 means that both first and second order parametric derivates of the two curve sections are proportional at their boundary. In G^2 curvature of the two successive curve sections will match at the joining position

Note:

1) **The joining point on the curve with respect to the parameter based on second derivates of Q(t) is the acceleration. While the Q'(t) is an tangent vector stating velocity. i.e.,** the tangent vector gives the velocity along the curve . the camera velocity and acceleration at join point should be continuous, to avoid jerky movements in the resulting animation sequence.

2) **Curve segment having a continuity C^1 implies the** G^1 **continuity** but the converse is generally not true. That is, G^1 curves are less restrictive than the C^1 , so curves can be G^1 but not necessarily C^1 .

Curves such as Spline curve, cubic spline curve are curve fitting method used to produce a smooth curve given a set of points throughout out the path weight are distributed. Spline is used in graphics to specify animation paths, to digitize drawings for computer storage mainly CAD application, use them for automobile bodies design in aircraft design, etc.

Spline curve is constructed using Control points which control the shape of the curve Spline curve is a composite curve formed with sections which are polynomial in nature. These pieces are joined together in such a manner that continuity condition at boundary is maintained. A piecewise parametric polynomial section when passes through each control point curve is known to **Interpolate** the set of control points refer *Figure 15*. **On the other hand when polynomial is not fitted to the set to very control point it is known to approximate the set of control points.**

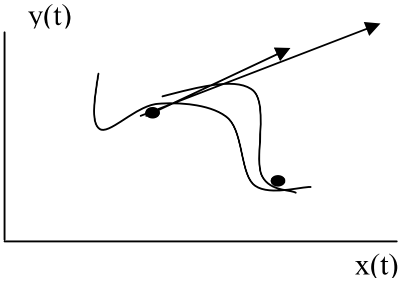


Figure 15 (a): Interpolating curve

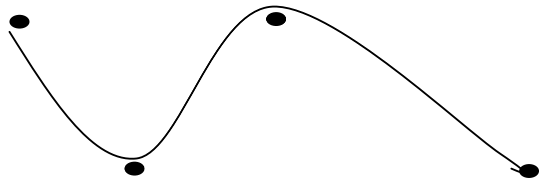


Figure 15 (b): Approximating Curve

Note: if P (u)→ = Bezier curve of order n and Q (u) → Bezier curve of order m

Then Continuities between P(u) and Q(u) are:

1) Positional continuity of 2 curves

   $P(u) = \sum_{i=0}^{n} p_i B_{n,i}(u)$ & $Q(u) = \sum_{j=0}^{m} q_j B_{m,j}(u)$

   is $p_n = q_0$

2) C^1 continuity of 2 curve P (u) & Q (u) says that point p_{n-1}, p_n on curve P(u) and q_0, q_1 points on curve Q(u) are collinear i.e.,

$$n(p_n - p_{n-1}) = m(q_1 - q_0)$$

$$q_1 = q_0 + \frac{n}{m}(p_n - p_{n-1})$$

$$\Rightarrow \frac{dp}{du}\bigg|_{u=1} = \frac{dq}{dv}\bigg|_{v=0}$$

   $G^{(1)}$ continuity of 2 curves P(u) & Q(u) at the joining namely the end of P(u) with the beginning of q(u) is:

$$p_n = q_0$$

$$n(p_n - p_{n-1}) = kn(q_1 - q_0), \text{ where k is a constant \& k > 0}$$

$$\Rightarrow p_{n-1}, p_n = q_0, q_1 \text{ are collinear}$$

3) c2 continuity:

   a) $C^{(1)}$ continuity
   b) $m(m-1)(q_0 - 2q_1 + q_2)$
      $= n(n-1)(p_n - 2p_{n-1} + p_{n-2})$

   i.e., points p_{n-2}, p_{n-1}, p_n of P(u) and points q_0, q_1, q_2 of Q(u) must be collinear

   further we can check whether both first and second order derivatives of two curve sections are the same at the intersection or not i.e.,

$$\frac{dp}{du}\bigg|_{u=1} = \frac{dq}{dv}\bigg|_{v=0}$$

   and

$$\frac{d^2p}{du^2}\bigg|_{u=1} = \frac{d^2q}{dv^2}\bigg|_{v=0}$$

   if they are same we can say we have C^2 continuity

Note: similarly we can define higher order parametric continuities

Example 8: An animation shows a car driving along a road which is specified by a Bezier curve with the following control points:

X_k	0	5	40	50
Y_k	0	40	5	15

The animation lasts 10 seconds and the key frames are to be computed at 1 second intervals. Calculate the position of the car on the road at the start of the 6th second of the animation.

Solution: Using similar methods to the previous exercise we can calculate the blending functions as:

1. $B_{03} = 3!/(0! \times (3-0)!)\ u^0(1 - u)^{(3-0)} = 1u^0(1 - u)^3 = \mathbf{(1 - u)^3}$
2. $B_{13} = 3!/(1! \times (3-1)!)\ u^1(1 - u)^{(3-1)} = 3u^1(1 - u)^2 = \mathbf{3u(1 - u)^2}$
3. $B_{23} = 3!/(2! \times (3-2)!)\ u^2(1 - u)^{(3-2)} = 3u^2(1 - u)^1 = \mathbf{3u^2(1 - u)}$
4. $B_{33} = 3!/(3! \times (3-3)!)\ u^3(1 - u)^{(3-3)} = 1u^3(1 - u)^0 = \mathbf{u^3}$

The function x(u) is equal to $x(u) = \sum x_k B_k$ where k=0,1,2,3

$$x(u) = \sum x_k B_k = x_0 B_{03} + x_1 B_{13} + x_2 B_{23} + x_2 B_{33}$$
$$= (0)(1 - u)^3 + 5\ [3u(1 - u)^2] + 40\ [3u^2(1 - u)] + 50\ u^3$$
$$= 15u(1 - u)^2 + 120u^2(1 - u) + 50u^3$$

similarly
$$y(u) = y_0 B_{03} + y_1 B_{13} + y_2 B_{23} + y_2 B_{33}$$
$$= (0)(1 - u)^3 + 40\ [3u(1 - u)^2] + 5\ [3u^2(1 - u)] + 15\ u^3$$
$$= 120u(1 - u)^2 + 15u^2(1 - u) + 15u^3$$

At the start of the sixth second of the animation, i.e., when u=0.6, we can use these equations to work out that x(0.6) = **29.52** and y(0.6) = **16.92**.

The path of the car looks like this

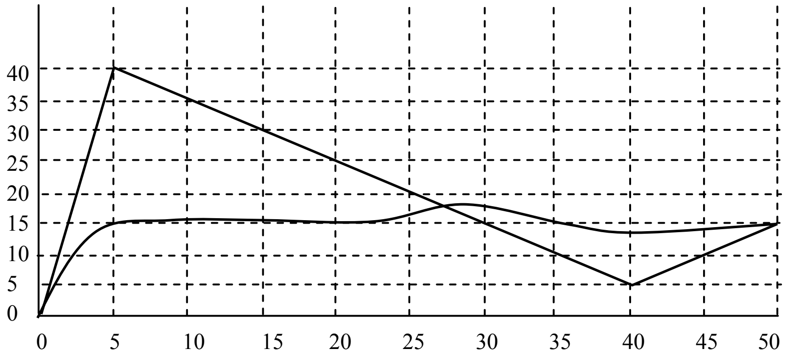


Figure 16

1.4.3 Bezier Surfaces

Two sets of Bezier curve can be used to design an object surface by specifying by an input mesh of control points. The Bézier surface is formed as the cartesian product of the blending functions of two Bézier curves.

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} B_{j,m}(v) B_{k,n}(u)$$

with $p_{j,k}$ specifying the location of the (m + 1) by (n + 1) control points.

The corresponding properties of the Bézier curve apply to the Bézier surface.

The surface does not in general pass through the control points except for the corners of the control point grid.

The surface is contained within the convex hull of the control points.

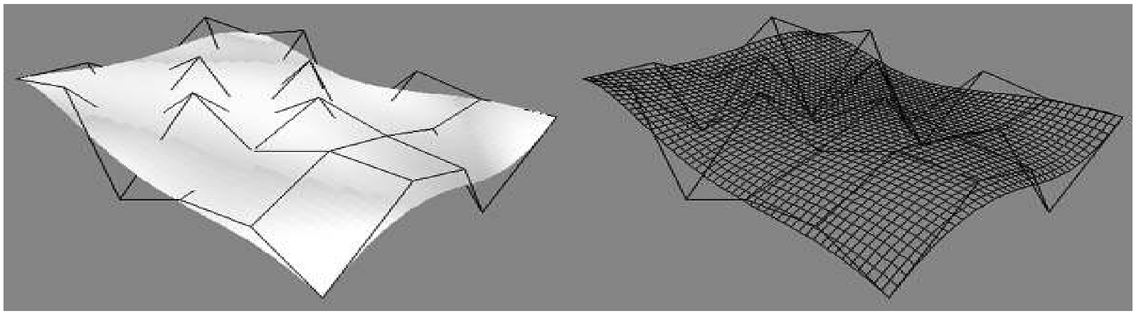


Figure 17

The control points are connected via a dashed line, and solid lenes shows curves of constant u and constant v. Each curve is ploted by varying v over the interval from 0 to 1, with u fixed at one of the values in this unit interval curves of constant v are plotted.

Figures (a), (b), (c) illustrate Bezier surface plots. The control points are connected by dashed lines, and the solid lines show curves of constant *u* and constant *v*. Each curve of constant *u* is plotted by varying *v* over the interval from 0 to 1, with u fixed at one of the values in this unit interval. Curves of constant *v* are plotted similarly.

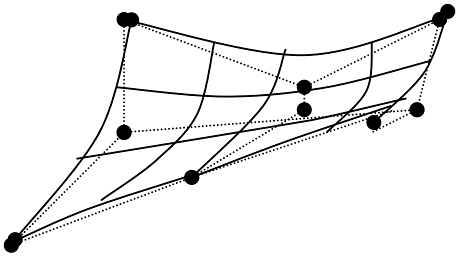


Figure 18 (a)

Bezier surfaces constructed for m=3, n=3, Dashed lines connect the control points

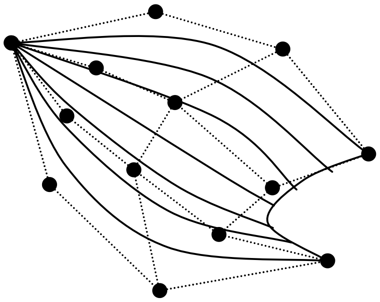


Figure 18 (b)

Bezier surfaces constructed for m=4, n=4. Dashed lines connect the control points.

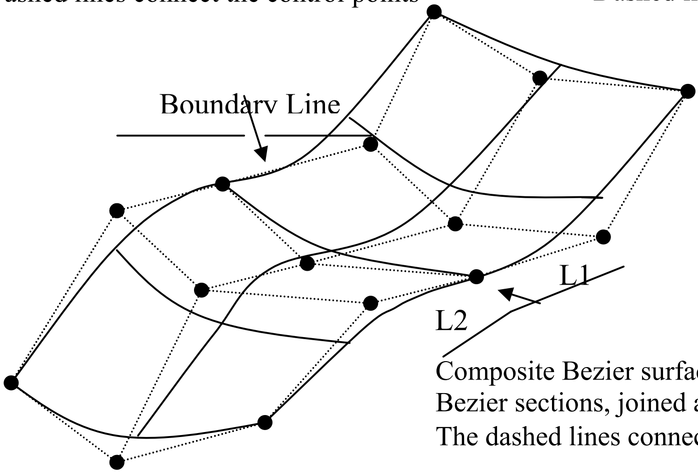


Figure 18 (c)

Composite Bezier surface constructed with two Bezier sections, joined at the indicated boundary line. The dashed lines connect specified control points.

In *Figure (c)* first-order continuity is established by making the ratio of length L_1 to length L_2 constant for each collinear line of control points across the boundary between the surface sections. *Figure (c)* also illustrates a surface formed with two Bezier sections. As with curves, a smooth transition from one section to the other is assured by establishing both zero-order and first-order continuity at the boundary line. Zero-order continuity is obtained by matching control points at the boundary. First-

order continuity is obtained by choosing control points along a straight line across the boundary and by maintaining a constant ration of collinear line segments for each set of specified control points across section boundaries.

☞ **Check Your Progress 3**

1) Based on the Bezier curve definition, derive the equation of the 3 point Bezier curve defined by the following control points. (-1,0), (0,2), and (1,0).

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2) Discuss how a Bezier surface is created. In particular discuss
   (a) The basic structure of a Bezier surface.
   (b) What blending function is used.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

1.5 SURFACE OF REVOLUTION

In the above sections we have learned various techniques of generating curves, but if we want to generate a close geometry, which is very symmetric in all the halves, i.e., front back, top, bottom; then it will be quite difficult for any person by doing it separately for each half. Under such a situation the concept of surface revolution is quite useful, because it helps in producing graphic objects such that they are very symmetric in every half. These geometries produced after Surface revolution are also known as Sweep Representations, which is one of the solid modeling construction techniques provided by many graphic packages. Sweep representations are useful for constructing three-dimensional objects that possess translational, rotational, or other symmetries. We can represent such objects by specifying a two-dimensional shape and a sweep that moves the shape through a region of space. A set of two-dimensional primitives, such as circles and rectangles, can be provided for sweep representations as menu options. Other methods for obtaining two-dimensional figures include closed spline-curve constructions and cross-sectional slices of solid objects. Let us discuss this simple technique of generating symmetric objects.

Say we are having a point P(x1,y1) in the X-Y plane (not at origin); if we make that point to revolve about Z axis then the point will trace circular geometry. Similarly, if the basic curve subjected to revolution is a Line segment then it will produce a

cylinder (symmetric about the axis of revolution). Therefore, different base curves we will get different surfaces of revolution, e.g.,

i) ***Base Curve*****:** say just a point when it rotates about x axis it will trace a circular surface of revolution.

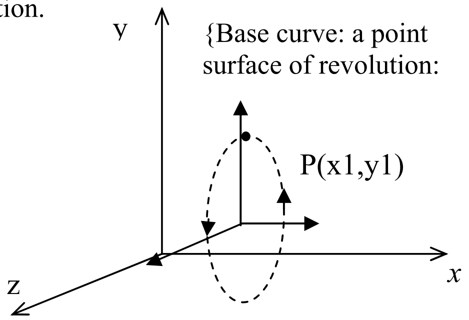


Figure 19

ii) If base curve ⇒ a line segment parallel to the *x* axis then a cylinder will be traced as a surface of revolution.

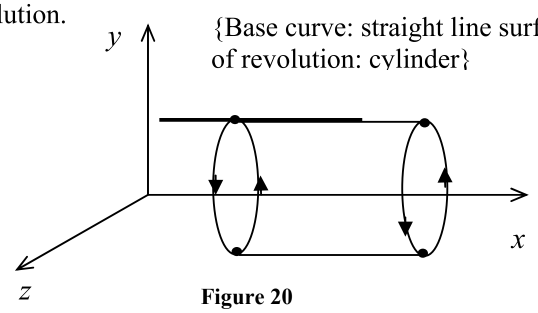


Figure 20

iii) If the line segment is inclined with one end point at the origin then a cone will be traced

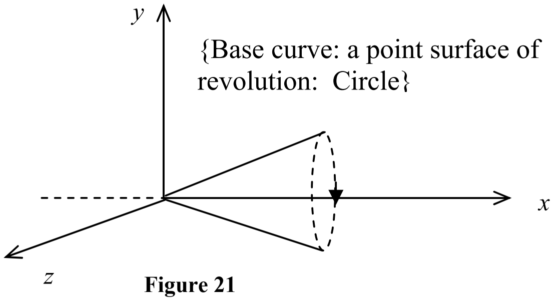


Figure 21

iv) If a semi-circle is rotated then a sphere is traced.

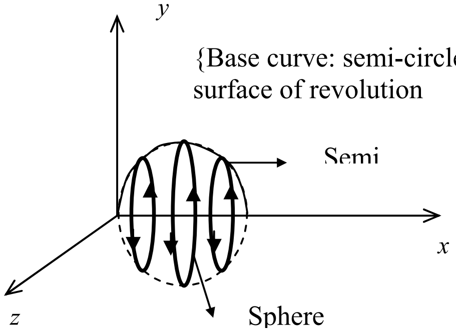


Figure 22

By this moment I hope it will be clear to you how a simple revolution of curve about any axis simplifies the creation of symmetric geometries You may use combination of basic curves to generate complex geometries. Now let us discuss the mathematics behind such creations.

How to find surface of revolution: Say there is an arbitrary base curve which when rotated about x-axis traces an arbitrary surface of revolution about the axis it is revolving

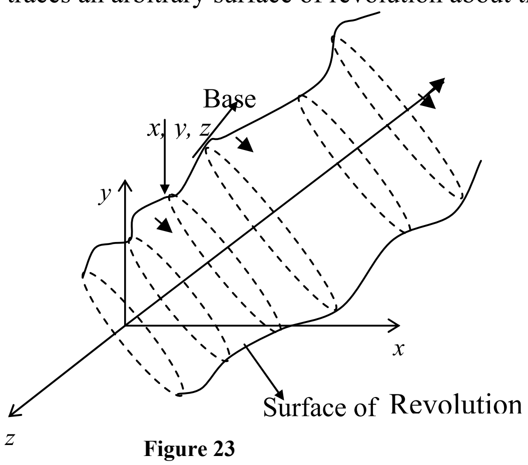


Figure 23

To find surface of revolution consider a point. (x, y, z) on base curve. Say curve is to revolve about x-axis $\Rightarrow$ Rotational transformation about x-axis is to be applied to (x, y, z).

$$(x, y, z) \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}; 0 \le \theta \le 2\pi.$$

As the revolution is about X axis so it has to be constant, for each section of the complete curve (you may change this value of X in steps, as you do in any programming language 'X++'; hence a continuous, symmetric curve of any arbitrary shape will be available).

$$\Rightarrow (x, y, z) \rightarrow (x, y, 0) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}; 0 \le \theta \le 2\pi.$$

$\Rightarrow (x, y, z) \rightarrow (x, y \cos\theta, y \sin \theta)$. These points trace surface of revolution about x-axis.

Note: a) if a point on base curve is given by parametric form, i.e., $(x(u), y(u), z(u))$ then, surface of revolution about x-axis will be
$[x(u), y(u), z(u)] \rightarrow [x(u), y(u) \cos \theta, y(u) \sin \theta]$
$0 \le u \le 1$; $0 \le \theta \le 2\pi$.

b) Tracing an image involves movement of points from one place to another, i.e., translational transformation is to be used. If (x, y, z) is a point on a base curve then moving the respective points on base curve from one place to other traces an image.

c) If $\vec{d}$ $\Rightarrow$ the direction in which curve is to be moved and
v $\Rightarrow$ scalar quantity representing the amount by which curve is to be shifted.
Displacing the curve by amount $v\vec{d}$, the curve will be traced at a new position or is swept to a new position.

$(x(u),\ y(u),\ z(u)) \rightarrow$ coordinate points of base curve in parametric form ($u \rightarrow$parameter)

$$(x(u), y(u), z(u)) \rightarrow (x(u), y(u), z(u)\ ) + \mathrm{v}\,\vec{d}$$

$$0 \le u \le 1; \qquad 0 \le v \le 1.$$

In general, we can specify sweep constructions using any path. For rotational sweeps, we can move along a circular path through any angular distance from 0 to 360°. For noncircular paths, we can specify the curve function describing the path and the distance of travel along the path. In addition, we can vary the shape or size of the cross section along the sweep path. Or we could vary the orientation of the cross section relative to the sweep path as we move the shape through a region space.

Figure 24 illustrates construction of a solid with a translational sweep. Translating the control points of the periodic spline curve in (a1) generates the solid shown in (b1), whose surface can be described with point function P(*u, v*).

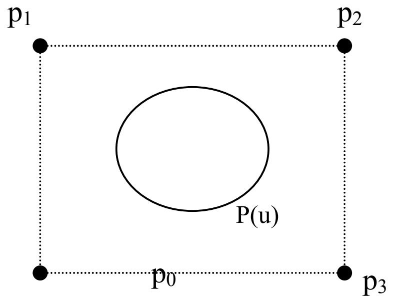


Figure 24 (a)

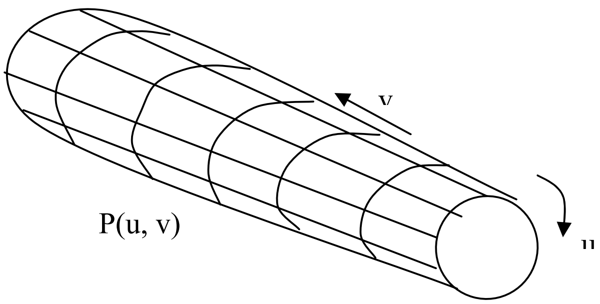


Figure 24 (b)

Figure 25 illustrates Construction of a solid with a rotational sweep. Rotating the control pints of the periodic spline curve in (a2) about the given rotation axis generates the solid shown in (b2), whose surface can be described with point function P(*u, v*).

Axis of Rotation

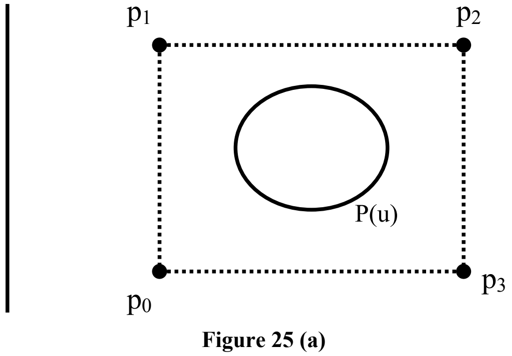


Figure 25 (a)

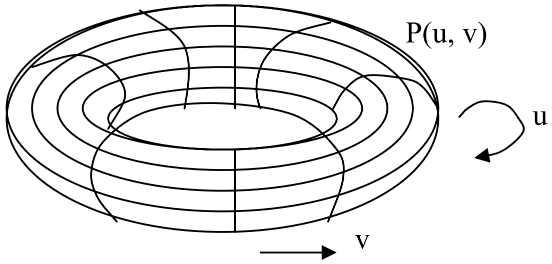


Figure 25 (b)

In the example of object shown in *Figure 25*, the designing is done by using a rotational sweep. This time, the periodic spline cross-section is rotated about an axis of rotation specified in the plane of the cross-section to produce the wireframe.

Any axis can be chosen for a rotational sweep. If we use a rotation axis perpendicular to the plane of the spline cross-section in *Figure 25(b)* we generate a two-dimensional shape. But if the cross section shown in this figure has depth, then we are using one three-dimensional object to generate another.

1.6 SUMMARY

This unit has covered the techniques of generating polygons, curves and closed surfaces. Under those techniques we have discussed various polygonal representational methods like tables, equations. Under the section of curves and surfaces after having brief discussion on mathematical representations of the curve through polynomial and parametric equation we have discussed the topic of Bezier curves and their applications. In the end of the unit we have discussed the concept of surface of revolution, which helps in generating 2D and 3D surfaces from 1D and 2D geometries.

1.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) So far as the accomplishment of the task to draw a polygon surface is concerned, it can be done with the help of Vertex table and Edge table but then there is a possibility that some edges can be drawn twice. Thus, the system will be doing redundant processing for the display of object. Hence the time of execution of task increases.

2) This extra information will help in rapid identification of the common edges between the polygon surfaces. This information in turn provides useful support in the process of surface rendering, which can be done smoothly when surface shading varies smoothly across the edge from one polygon surface to the next.

3) Yes, we can expand the vertex table, the additional information in the table will help in cross-referencing the vertices with corresponding edges. This will help in reducing he redundant processing of a vertex information associated with a polygon surface, hence reduces the execution time.

4) The more the information, the easier to check the error, if precisely all information is available in the table then you need not waste time in calculations and other tasks to diagnose an error in the polygon representation.

5) Left for the student to do.

Check Your Progress 2

1) Left as an exercise.

2) Left as an exercise.

Check Your Progress 3

1) To make the maths easier to understand we will first calculate the valuess of the blending function B_{02}, B_{12} and B_{22}.

$$B_{02} = 2!/(0! \times (2-0)!)\, u^0(1-u)^2 = 1u^0(1-u)^2 = \mathbf{(1-u)^2}$$

$$B_{12} = 2!/(0! \times (2-1)!)\, u^1(1-u)^{2-1} = 2u^1(1-u)^1 = \mathbf{2u(1-u)}$$

$$B_{22} = 2!/(0! \times (2-2)!)\, u^2(1-u)^{2-2} = 1u^2(1-u)^0 = \mathbf{u^2}$$

We can therefore obtain the following equations.

$$x(u) = \sum x_k B_k = x_0B_{02} + x_1B_{12} + x_2B_{22}$$
$$= (-1)(1-u)^2 + 0\,[2u(1-u)] + 1u^2$$
$$= 2u - 1 \qquad \text{similarly } y(u)$$

$$y(u) = \sum y_k B_k = y_0B_{02} + y_1B_{12} + y_2B_{22}$$
$$= (0)(1-u)^2 + 2\,[2u(1-u)] + 0u^2$$
$$= 4u(1-u)$$

2) a) A Bezier surface is a surface formed by two orthogonal Bezier curves such that they form a mesh of control points. Computing the value of each curve for a constant value of u, will produce the control points for the columns run across the curves.

b) The same blending function is used for Bezier surfaces that is used for Bezier splines. The equation for a surface is $P(u,v) = \sum_{j=0}^{m}\sum_{k=0}^{n} p_{j,k} B_{j,m}(v) B_{k,n}(u)$

UNIT 2 VISIBLE-SURFACE DETECTION

Structure **Page Nos.**

2.0 Introduction 33
2.1 Objectives 35
2.2 Visible-Surface Detection 35
  2.2.1 Depth Buffer (or z-buffer) Method 36
  2.2.2 Scan-Line Method 40
  2.2.3 Area-Subdivision Method 43
2.3 Summary 47
2.4 Solutions / Answers 48

2.0 INTRODUCTION

Given a set of 3-D objects and a viewing position. For the generation of realistic graphics display, we wish to determine which lines or surfaces of the objects are visible, either from the COP (for perspective projections) or along the direction of projection (for parallel projections), so that we can display only the visible lines or surfaces. For this, we need to conduct visibility tests. Visibility tests are conducted to determine the surface that is visible from a given viewpoint. This process is known as *visible-line* or *visible-surface determination*, or *hidden-line* or *hidden-surface elimination.*

To illustrate the concept for eliminating hidden-lines, edges or surfaces, consider a typical wire frame model of a cube (see *Figure 1*). A wire frame model represents a 3-D object as a line drawing of its edges. In *Figure 1(b)*, the dotted line shows the edges obscured by the top, left, front side of the cube. These lines are removed in *Figure 1(c)*, which results in a realistic view of the object. Depending on the specified viewing position, particular edges are eliminated in the graphics display. Similarly, *Figure 2(a)* represents more complex model and *Figure 2(b)* is a realistic view of the object, after removing hidden lines or edges.

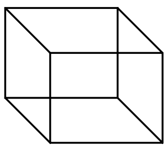


Figure1 (a)

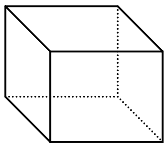


Figure1 (b)

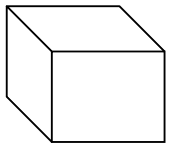


Figure 1(c)

.

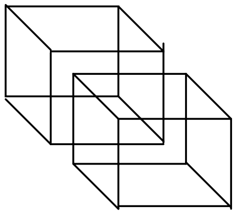


Figure 2(a)

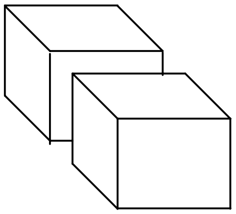


Figure 2(b)

There are numerous algorithms for identification of visible objects for different types of applications. Some methods require more memory, some involve more processing time, and some apply only to special types of objects. Deciding upon a method for a

particular application can depend on such factors as the complexity of the scene, type of objects to be displayed, available equipment, and whether static or animated displays are to be generated. These requirements have encouraged the development of carefully structured visible surface algorithms.

There are two fundamental approaches for visible-surface determination, according to whether they deal with their projected images or with object definitions directly. These two approaches are called *image-space approach* and *object-space approach*, respectively. Object space methods are implemented in the physical coordinate system in which objects are defined whereas image space methods are implemented in screen coordinate system in which the objects are viewed.

In both cases, we can think of each object as comprising one or more polygons (or more complex surfaces). The first approach (image-space) determines which of n objects in the scene is visible at each pixel in the image. The pseudocode for this approach looks like as:

for*(each pixel in the image)*
{
determine the object closest to the viewer that is passed by the projector through the pixel;
draw the pixel in the appropriate color;
}

This approach requires examining all the objects in the scene to determine which is closest to the viewer along the projector passing through the pixel. That is, in an image-space algorithm, the visibility is decided point by point at each pixel position on the projection plane. If the number of objects is 'n' and the pixels is 'p' then effort is proportional to n.p.

The second approach (object-space) compares all objects directly with each other within the scene definition and eliminates those objects or portion of objects that are not visible. In terms of pseudocode, we have:

for *(each object in the world)*
{
determine those parts of the object whose view is unobstructed (not blocked) by other
parts of it or any other object;
draw those parts in the appropriate color;
}

This approach compares each of the n objects to itself and to the other objects, and discarding invisible portions. Thus, the computational effort is proportional to n^2.

Image-space approaches require two buffers: one for storing the pixel intensities and another for updating the depth of the visible surfaces from the view plane.

In this unit, under the categories of image space approach, we will discuss two methods, namely, *Z-buffer (or Depth-buffer) method* and *Scan-line method*. Among all the algorithms for visible surface determination, the Depth-buffer is perhaps the simplest, and is the most widely used. *Z-buffer method*, detects the visible surfaces by comparing surface depths (z-values) at each pixel position on the projection plane. In *Scan-line method*, all polygon surfaces intersecting the scan-line are examined to determine which surfaces are visible on the basis of depth calculations from the view plane. For scenes with more than one thousand polygon surfaces, Z-buffer method is the best choice. This method has nearly constant processing time, independent of

number of surfaces in a scene. The performance of *Z*-buffer method is low for simple scenes and high with complex scenes. Scan-line methods are effectively used for scenes with up to thousand polygon surfaces.

The third approach often combines both object and image-space calculations. This approach utilizes depth for sorting (or reordering) of surfaces. They compare the depth of overlapping surfaces and identify one that is closer to the view-plane. The methods in this category also use image-space for conducting visibility tests.

Area-subdivision method is essentially an image-space method but uses object-space calculations for reordering of surfaces according to depth. The method makes use of area coherence in a scene by collecting those areas that form part of a single surface. In this method, we successively subdivide the total viewing area into small rectangles until each small area is the projection of part of a single visible surface or no surface at all.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- understand the meaning of Visible-surface detection;
- distinguish between image-space and object-space approach for visible-surface determination;
- describe and develop the depth-buffer method for visible-surface determination;
- describe and develop the Scan-line method for visible-surface determination, and
- describe and develop the Area-Subdivision method for visible-surface determination.

2.2 VISIBLE-SURFACE DETECTION

As you know for the generation of realistic graphics display, hidden surfaces and hidden lines must be identified for elimination. For this purpose we need to conduct visibility tests. Visibility tests try to identify the visible surfaces or visible edges that are visible from a given viewpoint. Visibility tests are performed by making use of either i) *object-space* or ii) *image-space* or iii) both *object-space* and *image-spaces*.

Object-space approaches use the directions of a surface normal w.r.t. a viewing direction to detect a back face. *Image-space* approaches utilize two buffers: one for storing the pixel intensities and another for updating the depth of the visible surfaces from the view plane. A method, which uses both *object-space* and *image-space,* utilizes depth for sorting (or reordering) of surfaces. The methods in this category also use image-space for conducting visibility tests. While making visibility tests, coherency property is utilized to make the method very fast.

In this section, we will discuss three methods (or algorithms) for detecting visible surfaces:

- Depth-buffer method
- Scan-line method
- Area subdivision method

Depth-buffer method and Scan-line method come under the category of image-space, and area-subdivision method uses both object-space and image-space approach.

2.2.1 Depth-buffer (or z-buffer) Method

Depth-buffer method is a fast and simple technique for identifying visible-surfaces. This method is also referred to as the *z*-buffer method, since object depth is usually measured from the view plane along the z-axis of a viewing system. This algorithm compares surface depths at each pixel position (x,y) on the view plane. Here we are taking the following assumption:

- Plane of projection is z=0 plane
- Orthographic parallel projection.

For each pixel position (x,y) on the view plane, the surface with the smallest z-coordinate at that position is visible. For example, *Figure 3* shows three surfaces S1, S2, and S3, out of which surface S1 has the smallest z-value at (x,y) position. So surface S1 is visible at that position. So its surface intensity value at (x,y) is saved in the refresh-buffer.

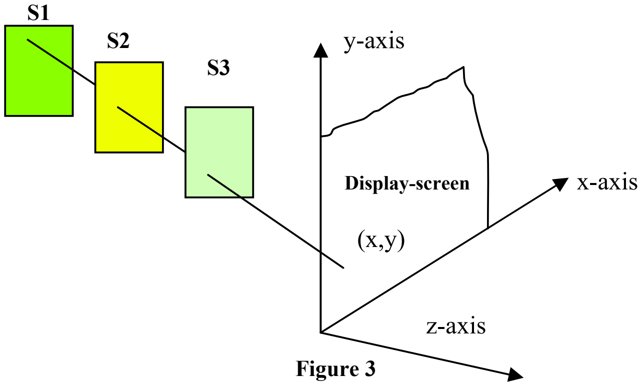


Figure 3

Here the projection is orthographic and the projection plane is taken as the xy-plane. So, each (x,y,z) position on the polygon surfaces corresponds to the orthographic projection point (x,y) on the projection plane. Therefore, for each pixel position (x,y) on the view plane, object depth can be compared by comparing z-values, as shown in *Figure 3*.

For implementing z-buffer algorithm two buffer areas (two 2-D arrays) are required.

1) Depth-buffer: **z-buffer**(i,j) , to store z-value, with least z, among the earlier z-values for each (x,y) position on the view plane.
2) Refresh-buffer: **COLOR**(i,j): for storing intensity values for each position.

We summarize the steps of a depth-buffer algorithm as follows:

Given: A list of polygons {P1,P2,…..,Pn}.
Step1: Initially all positions (x,y) in the depth-buffer are set to 1.0 (maximum depth) and the refresh-buffer is initialized to the background intensity i.e.,
__z-buffer__(x,y):=1.0; and
__COLOR__(x,y):= Background color.

Step2: For each position on each polygon surface (listed in the polygon table) is then processed (scan-converted), one scan line at a time. Calculating the depth (z-value) at each (x,y) pixel position. The calculated depth is then compared to the value previously stored in the depth buffer at that position to determine visibility.

a) If the calculated z-depth is less than the value stored in the depth-buffer, the new depth value is stored in the depth-buffer, and the surface intensity at

that position is determined and placed in the same (x,y) location in the refresh-buffer, i.e.,

If z-depth< ***z-buffer***(x,y), then set
z-buffer*(x,y)=z-depth;*
COLOR*(x,y)=*$I_{surf}(x,y)$; // where $I_{surf}(x,y)$ is the projected intensity value of the polygon
surface Pi at pixel position (x,y).

After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the refresh-buffer contains the corresponding intensity values for those surfaces.

In terms of pseudo code, we summarize the depth-buffer algorithm as follows:

Given: *A list of polygons {P1,P2,…..,Pn}*
Output: A COLOR array, which display the intensity of the visible polygon surfaces.

```
Initialize:
        z-buffer(x,y):=0; and
        COLOR(x,y):= Back-ground color.
Begin
    For (each polygon P in the polygon list) do {
      For (each pixel (x,y) that intersects P) do {
        Calculate z-depth of P at (x,y)
        If (z-depth < z-buffer[x,y]) then {
          z-buffer(x,y)=z-depth;
          COLOR(x,y)=Intensity of P at (x,y);
        }
      }
    }
display COLOR array.
```

Calculation of depth values, z, for a surface position (x,y):

We know that for any polygon faces, the equation of the plane is of the form:

$A.x+B.y+C.z+D=0$ --------------------(1) , where A, B, C, D are known to us.

To calculate the depth value z, we have to solve the plane equation (1) for z:

$z=(-A.\,x - B\,.\,y - D)/C$ --------(2)

Consider a polygon in *Figure 4* intersected by scan-lines at y and y – 1 on y-axis.

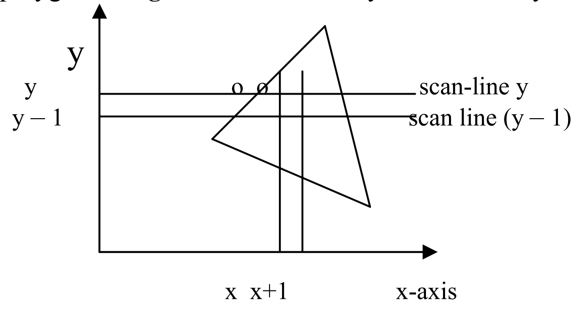


Figure 4

Now, if at position (x, y) equation (2) evaluates to depth z, then at next position (x+1,y) along the scan line, the depth z_H can be obtained as:

$z_H = [-A.(x+1) - B.y - D]/C$ --------------(3)

From equation (2) and (3), we have

$z - z_H = A/C$

$z_H = z - A/C$ -----------------(4)

The ratio –A/C is constant for each surface. So we can obtain succeeding depth values across a scan-line from the preceding values by a single addition. On each scan-line, we start by calculating the depth on the left edge of the polygon that intersects that scan-line and then proceed to calculate the depth at each successive position across the scan -line by Equation-(4) till we reach the right edge of the polygon.

Similarly, if we are processing down, the vertical line x intersecting the (y–1)th scan-line at the point (x, y-1). Thus from Equation (2) the depth z_v is obtained as:

$z_v = [-A.x - B.(y-1) - D]/C$

$= ([-A.x - B.y - D]/C) + B/C$

$= z + B/C$ ----------------(5)

Starting at the top vertex, we can recursively calculate the x position down the left edge of the polygon from the relation: x'=x-1/m, where m is the slope of the edge (see *Figure 5*). Using this x position, the depth z' at (x',y-1) on the (y-1) scan-line is obtained as:

$z' = [-A.x' - B.(y-1) - D]/C$

$= [-A.(x-1/m) - B.(y-1) - D]/C$

$= z + (A/m + B)/C$ ---------------------(6)

Since m=∞ for a vertical line, Equation (6) becomes equation (5).

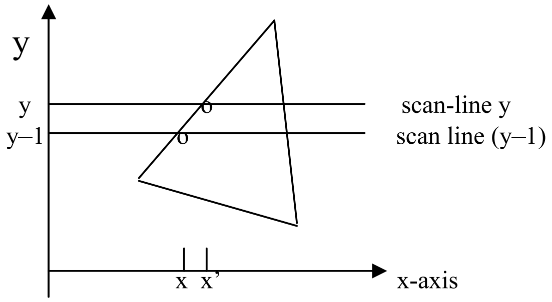


Figure 5: Intersection position on successive scan lines along a left polygon edge

Thus, if we are processing down, then we can obtain succeeding depth values across a scan-line from the preceding values by a single addition by using Equation (5), i.e., $z_v = z + B/C$.

Thus, the summary of the above calculations are as follows:

- You can obtain succeeding depth values across a scan-line from the preceding values by a single subtraction, i.e., z'=z–A/C .
- If we are processing down, then we can also obtain succeeding depth values across a scan-line from the preceding values by a single addition, i.e., z'= z+(A/m+B)/C . In other words, if we are processing up, then we can obtain succeeding depth values across a scan-line from the preceding values by a single subtraction, i.e., z'= z– (A/m+B)/C.

The following *Figure 6* summarizes the above calculations.

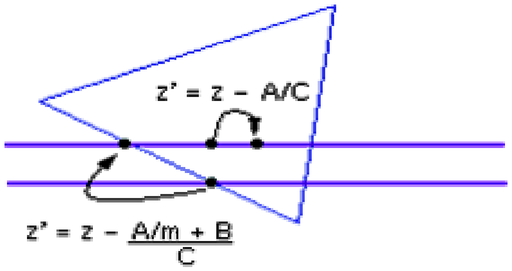


Figure 6: Successive depth values, when processing left to right or processing up across a scan-line

Advantages (z-buffer method):

1) The z-buffer method is easy to implement and it requires no sorting of surface in a scene.
2) In z-buffer algorithm, an arbitrary number of objects can be handled because each object is processed one at a time. The number of objects is limited only by the computer's memory to store the objects.
3) Simple hardware implementation.
4) Online algorithm (i.e., we dont need to load all polygons at once in order to run algorithm).

Disadvantages:

1) Doubles memory requirements (at least), one for z-buffer and one for refress-buffer.
2) Device dependent and memory intensive.
3) Wasted computation on drawing distant points that are drawn over with closer points that occupy the same pixel.
4) Spends time while rendering polygons that are not visible.
5) Requires re-calculations when changing the scale.

Example 1: How does the z-buffer algorithm determine which surfaces are hidden?

Solution: Z-buffer algorithm uses a two buffer area each of two-dimensional array, one z-buffer which stores the depth value at each pixel position (x,y), another frame-buffer which stores the intensity values of the visible surface. By setting initial values of the z-buffer to some large number (usually the distance of back clipping plane), the problem of determining which surfaces are closer is reduced to simply comparing the present depth values stored in the z-buffer at pixel (x,y) with the newly calculated depth value at pixel (x,y). If this new value is less than the present z-buffer value, this value replaces the value stored in the z-buffer and the pixel color value is changed to the color of the new surface.

Example 2: What is the maximum number of objects that can be handled by the z-buffer algorithm?

Solution: In z-buffer algorithm, an arbitrary number of objects can be handled because each object is processed one at a time. The number of objects is limited only by the computer's memory to store the objects.

Example 3: What happens when two polygons have the same z value and the z-buffer algorithm is used?

Solution: z-buffer algorithms, changes colors at a pixel if $z(x,y)<z_{buf}(x,y)$, the first polygon surface will determine the color of the pixel.

Example 4: Assume that one allow 256 depth value level to be used. Approximately how many memory would a 512x512 pixel display require to store z-buffer?

Solution: A system that distinguishes 256 depth values would require one byte of memory ($2^8=256$) to represent z-value.

☞ Check Your Progress 1

1) z-buffer method use(s) ………….. :
   a) Only object-space approach b) Only image-space approach c) both object-space & Image-space.

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

2) What happens when two polygons have the same z value and the z-buffer algorithm is used?

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

3) Assuming that one allows 2^{32} depth value levels to be used, how much memory would a 1024x768 pixel display require to stores the z-buffer?

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

2.2.2 Scan-Line method

In contrast to z-buffer method, where we consider one surface at a time, scan-line method deals with multiple surfaces. As it processes each scan-line at a time, all polygon intersected by that scan-line are examined to determine which surfaces are visible. The visibility test involves the comparison of depths of each overlapping surface to determine which one is closer to the view plane. If it is found so, then it is declared as a visible surface and the intensity values at the positions along the scan-line are entered into the refresh-buffer.

Assumptions:
1. Plane of projection is Z=0 plane.
2. Orthographic parallel projection.

3. Direction of projection, d = (0,0, −1)
4. Objects made up of polygon faces.

Scan-line algorithm solves the hidden- surface problem, one scan-line at a time, usually processing scan lines from the bottom to the top of the display.

The scan-line algorithm is a one-dimensional version of the depth –Buffer. We require two arrays, intensity [x] & depth [x] to hold values for a single scan-line.

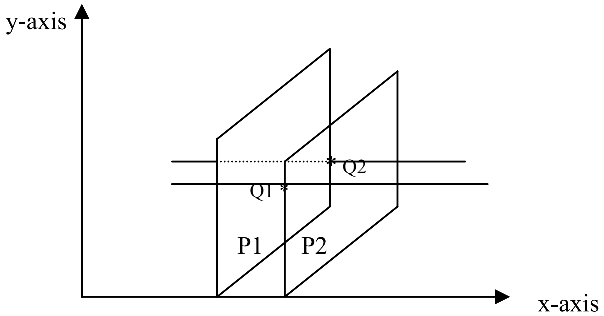


Figure 7

Here at $Q_1$and Q_2 both polygons are active (i.e., sharing).

Compare the z-values at Q_1 for both the planes (P_1 & P_2). Let $z_1^{(1)}$,$z_1^{(2)}$ be the z-value at Q_1 ,corresponding to P_1& P_2 polygon respectively.

Similarly
$z_2^{(1)}$, $z_2^{(2)}$ are the z-values at Q_2, corresponding to P_1 & P_2 polygon respectively.

Case1: $\left.\begin{array}{l} z_1^{(1)} < z_1^{(2)} \\ z_2^{(1)} < z_2^{(2)} \end{array}\right\}$ ➔ Q1,Q2 is filled with the color of P2.

Case2: $\left.\begin{array}{l} z_1^{(2)} < z_1^{(1)} \\ z_2^{(2)} < z_2^{(1)} \end{array}\right\}$ ➔ Q1,Q2 is filled with the color of P2.

Case3: Intersection is taking place.

In this case we have to go back pixel by pixel and determine which plane is closer. Then choose the color of the pixel.

Algorithm (scan-line):

For each scan line perform step (1) through step (3).

1) For all pixels on a scan-line, set
   depth [x]=1.0 (max value) & Intensity [x] = background-color.

2) For each polygon in the scene, find all pixels on the current scan-line (say S1) that lies within the polygon. For each of these x-values:

   a) calculate the depth z of the polygon at (x,y)
   b) if z < depth [x], set depth [x]=z & intensity corresponding to the polygon's shading.

3) After all polygons have been considered, the values contained in the intensity array represent the solution and can be copied into a frame-buffer.

Advantages of Scan line Algorithm:

Here, every time, we are working with one-dimensional array, i.e., x[0…x_max] for color not a 2D-array as in depth buffer algorithm.

Example 5: Distinguish between z-buffer method and scan-line method. What are the visibility test made in these methods?

Solution: In z-buffer algorithm every pixel position on the projection plane is considered for determining the visibility of surfaces w. r. t. this pixel. On the other hand in scan-line method all surfaces intersected by a scan line are examined for visibility. The visibility test in z-buffer method involves the comparison of depths of surfaces w. r. t. a pixel on the projection plane. The surface closest to the pixel position is considered visible. The visibility test in scan-line method compares depth calculations for each overlapping surface to determine which surface is nearest to the view-plane so that it is declared as visible.

Example6: Given two triangles P with vertices P1(100,100,50), P2(50,50,50), P3(150,50,50) and q with vertices Q1(40,80,60), q2(70,70,50), Q3(10,75,70), determine which triangle should be painted first using the scan-line method.

Solution: In the scan-line method, two triangles P and Q are tested for overlap in xy-plane. Then they are tested for depth overlap. In this question, there is no overlap in the depth. But P and Q have overlap in xy-plane. So the Q is painted first followed by P.

☞ Check Your Progress 2

1) All the algorithm, which uses image-space approach, requires:
   a) One buffer-area b) two buffer-areas c) three-buffer areas

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

2) All the algorithm, which uses object-space approach, requires:
   a) One buffer-area b) two buffer-areas c) three- buffer areas

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

3) Scan line method deals with ______________ surface(s) at a time for ascertaining visibility
   a) single b) two c) multiple d) 100

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

4) What are the relative merits of object-space methods and image-space methods?

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

2.2.3 Area-Subdivision method

This method is essentially an image-space method but uses object-space operations reordering (or sorting) of surfaces according to depth. This method takes advantage of area-coherence in a scene by locating those view areas that represent part of a single surface. In this method we successively subdivide the total viewing (screen) area, usually a rectangular window, into small rectangles until each small area is the projection of part of a single visible surface or no surface at all.

Assumptions:

- Plane of projection is z=0 plane
- Orthographic parallel projection
- Direction of projection d=(0,0,-1)
- Assume that the viewing (screen) area is a square
- Objects are made up of polygon faces.

To implement the area-subdivision method, we need to identify whether the area is part of a single surface or a complex surface by means of visibility tests. If the tests indicate that the view is sufficiently complex, we subdivide it. Next, we apply the tests to each of the smaller areas and then subdivide further if the tests indicate that the visibility of a single surface is still uncertain. We continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel.

Starting with the full screen as the initial area, the algorithm divides an area at each stage into 4 smaller area, as shown in *Figure 8*, which is similar to quad-tree approach.

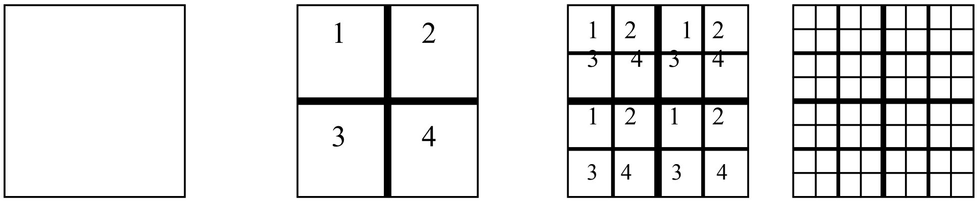


Figure 8 (a) Initial area (b) 1st subdivision (c) 2nd Subdivision (d) 3rd Subdivision

Test to determine the visibility of a single surface are made by comparing surfaces (i.e., polygons P) with respect to a given screen area A. There are 4 possibilities:

1) **Surrounding polygon:** Polygon that completely contains the area (*Figure 9(a)*).
2) **Intersecting (or overlapping) polygon:** Polygon that intersects the area (*Figure 9(b)*).
3) **Contained polygon:** polygon that is completely contained within the area (*Figure 9(c)*).
4) **Disjoint polygon:** Polygon that is completely outside the area (*Figure 9(d)*).

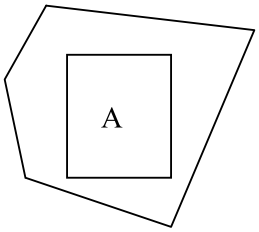


Figure 9(a)

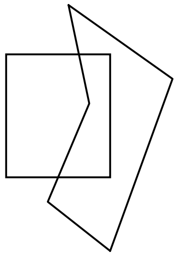


Figure 9(b)

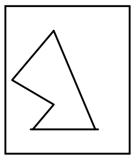


Figure 9(c)

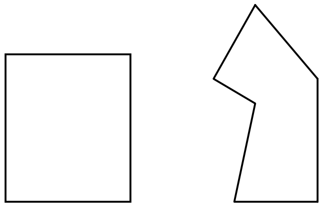


Figure 9(d)

The classification of the polygons within a picture is the main computational expense of the algorithm and is analogous to the clipping algorithms. With the use of any one of the clipping algorithms, a polygon in category 2 (intersecting polygon) can be clipped into a contained polygon and a disjoint polygon (see *Figure 10*). Therefore, we could proceed as if category 2 were eliminated.

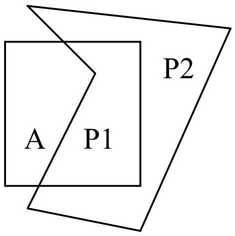


Figure 10

No further subdivisions of a specified area are needed, if one of the following conditions is true:

Case 1: All the polygons are disjoint from the area. In this case, the background color can be displayed in the area.

Case 2: Exactly one polygon faces, after projection, intersecting or contained in the square area. In this case the area is first filled with the background color, and then the part of the polygon contained in the area is scan converted.

Case 3: There is a single surrounding polygon, but no intersecting or contained polygons. In this case the area is filled with the color of the surrounding polygon.

Case 4: More than one polygon is intersecting, contained in, or surrounding the area, but one is a surrounding polygon that is in front of all the other polygons. Determining whether a surrounding polygon is in front is done by computing the z coordinates of the planes of all surrounding, intersecting and contained polygons at the four corners of the area; if there is a surrounding polygon whose four corner z coordinates are larger than one those of any of the other polygons, then the entire area can be filled with the color of this surrounding polygon.

To check whether the polygon is any one of these four cases, we have to perform the following test:

Test 1: For checking disjoint polygons (use **Min-max test**).
Suppose you have two polygons *P1* and *P2*. The given polygons *P1* and *P2* are disjoint if any of the following four conditions is satisfied (see *Figures-11(a) and 11(b)*): These four tests are called **Min-max test**.

i) $x^{(1)}_{max} < x^{(2)}_{min}$
ii) $x^{(2)}_{max} < x^{(1)}_{min}$
iii) $y^{(1)}_{max} < y^{(2)}_{min}$
iv) $y^{(2)}_{max} < y^{(1)}_{min}$

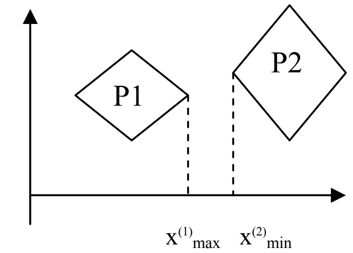


Figure 11(a)

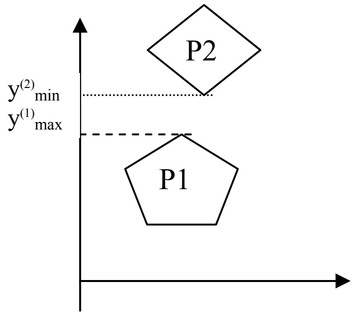


Figure 11(b)

Test 2: (Intersection Test)**:** If **Min-max test** fails then we go for intersection test. Here we take each edge one by one and see if it is intersecting. For example, see Figure 12, for each edge of *P1* we find the intersection of all edges of *P2*.

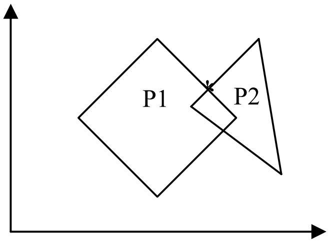


Figure 12

Test 3: (Containment test): If intersection test fails, then it can be either contained polygon or surrounding polygon. So we do the containment test. For this test we have the following three cases, shown in *Figures 13(a),(b) and (c)*.

a) P1 contains P2.
b) P2 contains P1.
c) P1 and P2 are disjoint.

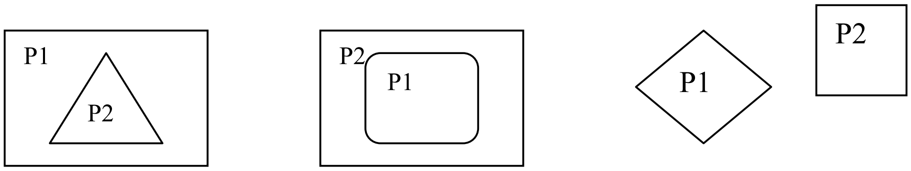


Figure 13(a): P1 contained P2 **(b) P2 contained P1** **(c) P1and P2 are disjoint**

Case a: Verify a vertex point of P2 lies inside of P1. If the result is true, P2 is completely inside of P1.

Case b: If the result of case-a is not true, then verify whether P2 contains a vertex point of P1. If the result is true, then P2 contains P1.

Case c: If both **case-a** and **case-b** (containment test) failed then we conclude that P1 and P2 are disjoint.

For a given screen area, we keep a potentially visible polygons list (PVPL), those in categories 1, 2 and 3. (Disjoint polygons are clearly not visible). Also, note that on subdivision of a screen area, surrounding and disjoint polygons remain surrounding and disjoint polygons of the newly formed areas. Therefore, only contained and intersecting polygons need to be reclassified.

Removing Polygons Hidden by a Surrounding Polygon:

The key to efficient visibility computation lies in the fact that a polygon is not visible if it is in back of a surrounding polygon. Therefore, it can be removed from the PVPL. To facilitate processing, this list is sorted by z_{min}, the smallest z coordinate of the polygon within this area. In addition, for each surrounding polygon *S,* we also record its largest z coordinate, z_{smax}.

If, for a polygon *P* on the list, $z_{pmin} > z_{smax}$ (for a surrounding polygon *S*), then *P* is hidden by *S* and thus is not visible. In addition, all other polygons after *P* on the list will also be hidden by *S*, so we can remove these polygons from the PVPL.

Subdivision Algorithm

1) Initialize the area to be the whole screen.

2) Create a PVPL w.r.t. an area, sorted on zmin (the smallest z coordinate of the polygon within the area). Place the polygons in their appropriate categories. Remove polygons hidden by a surrounding polygon and remove disjoint polygons.

3) Perform the visibility decision tests:

   a) If the list is empty, set all pixels to the background color.
   b) If there is exactly one polygon in the list and it is classified as intersecting (category 2) or contained (category 3), color (scan-converter) the polygon, and color the remaining area to the background color.
   c) If there is exactly one polygon on the list and it is a surrounding one, color the area the color of the surrounding polygon.
   d) If the area is the pixel (x,y), and neither a, b, nor c applies, compute the z coordinate z(x, y) at pixel (x, y) of all polygons on the PVPL. The pixel is then set to the color of the polygon with the smallest z coordinate.

4) If none of the above cases has occurred, subdivide the screen area into fourths. For each area, go to step 2.

Example 7: Suppose there are three polygon surfaces P,Q, R with vertices given by:
P: P1(1,1,1), P2(4,5,2), P3(5,2,5)
Q: Q1(2,2,0.5), Q2(3,3,1.75), Q3(6,1,0.5)
R: R1(0.5,2,5.5), R2(2,5,3), R3(4,4,5)

Using the Area subdivision method, which of the three polygon surfaces P, Q, R obscures the remaining two surfaces? Assume z=0 is the projection plane.

Solution: Here, we have z=0 is the projection plane and P, Q, R are the 3-D planes. We apply first three visibility decision tests i.e. (a), (b) and (c), to check the bounding rectangles of all surfaces against the area boundaries in the xy-plane. Using test 4, we can determine whether the minimum depth of one of the surrounding surface S is closer to the view plane.

Example 8: What are the conditions to be satisfied, in Area-subdivision method, so that a surface not to be divided further?

Solution: In an area subdivision method, the given specified area IS not to be divided further, if the following four conditions are satisfied:

1) Surface must be outside the specified area.

2) There must be one overlapping surface or one inside surface.
3) One surrounding surface but not overlapping or no inside surface.
4) A surrounding surface/obscures all other surfaces with the specified area.

☞ **Check Your Progress 3**

1) Area- subdivision method uses:
   a) Only image-space b) Only object-space c) both image and object space

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

2) What are the basic concepts of Area-subdivision method?

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

2.3 SUMMARY

- For displaying a realistic view of the given 3D-object, hidden surfaces and hidden lines must be identified for elimination.

- The process of identifying and removal of these hidden surfaces is called the *visible-line* or *visible-surface determination*, or *hidden-line* or *hidden-surface elimination.*

- To construct a realistic view of the given 3D object, it is necessary to determine which lines or surfaces of the objects are visible. For this, we need to conduct visibility tests.

- Visibility tests are conducted to determine the surface that is visible from a given viewpoint.

- There are two fundamental approaches for visible-surface determination, according to whether they deal with their projected images or with object definitions directly. These two approaches are called *image-space approach* and *object-space approach*, respectively.

- Object space methods are implemented in the physical coordinate system in which objects are defined whereas image space methods are implemented in screen coordinate system in which the objects are viewed.

- Image-space approach requires examining all the objects in the scene to determine which is closest to the viewer along the projector passing through the pixel. That is, the visibility is decided point by point at each pixel position on the projection plane. If the number of objects is ‘n’ and the pixels is ‘p’ then effort is proportional to **n.p.**

- Object-space approach compares all objects directly with each other within the scene definition and eliminates those objects or portion of objects that are not visible.

- Object-space approach compares each of the n objects to itself and to the other objects, discarding invisible portions. Thus the computational effort is proportional to n^2.

- Under the category of *Image space approach*, we have two methods: 1) *Z-buffer* method and 2) *Scan-line* method.

- Among all the algorithms for visible surface determination, the Z-buffer is perhaps the simplest, and is the most widely used method.

- *Z-buffer* method detects the visible surfaces by comparing surface depths (z-values) at each pixel position on the projection plane.

- For implementing z-buffer algorithm, we require two buffer areas (two 2-D arrays): 1) **Depth-buffer[i,j]**, to store the depth-value of the visible surface for each pixel in the view plane, and 2) **Refresh-buffer[i,j],** to store the pixel intensities of the visible surfaces.

- In contrast to z-buffer method, *Scan-line method* deals with multiple surfaces. As it processes each scan-line at a time, all polygon intersected by that scan-line are examined to determine which surfaces are visible. The visibility test involves the comparison of depths of each overlapping surfaces to determine which one is closer to the view plane. If it is found so, then it is declared as a visible surface and the intensity values at the positions along the scan-line are entered into the refresh-buffer.

- *Area-subdivision method* is essentially an image-space method but uses object-space calculations for reordering of surfaces according to depth. The method makes use of area coherence in a scene by collecting those areas that form part of a single surface. In this method, we successively subdivide the total viewing area into small rectangles until each small area is the projection of part of a single visible surface or no surface at all.

2.4 SOLUTIONS/ANSWERS

Check Your Progress 1

1) b

2) z-buffer algorithms, changes colors at a pixel if $z(x,y) < z_{buf}(x,y)$, the first polygon surface (which is written) will determine the color of the pixel.

3) A system that distinguishes 2^{32} depth values would require four bytes of memory to represent each z value. Thus total memory needed= 4x1024x768=3032K

Check Your Progress 2

1) b

2) a

3) c

4) Image space approaches we determine which of the objects in the scene is visible, at each pixel, by comparing the z-value of each object. Object-space approach determines the visibility of each object in the scene. For this all objects are compared within scene definition.

   Image-space methods are implemented in screen coordinate system whereas Object-space methods are implemented in the physical coordinate system.

   Image-space approaches were developed for raster devices whereas object-space approaches were developed for vector graphics systems. In case of image-space approaches, the results are crude and limited by the resolution of the screen whereas in object-space approaches, we have very precise results (generally to the precision of a machine).

Check Your Progress 3

1) a

2) The area-subdivision algorithm works as follows:

   Step-1: A polygon is seen from within a given area of the display screen if the projection of that polygon overlaps the given area.

   Step-2 : Of all polygons that overlap a given screen area, the one that is visible in this area is the one in front of all the others.

   Step-3 : If we cannot decide which polygon is visible (in front of the others) from a given region, we subdivide the region into smaller regions until visibility decisions can be made (even if we subdivide the region up to the pixel level).

UNIT 3 POLYGON RENDERING AND RAY TRACING METHODS

Structure **Page Nos.**

3.1	Introduction	50
3.2	Objectives	51
3.3	Illumination Model	51
	3.3.1 Ambient Reflection	55
	3.3.2 Diffuse Reflection	56
	3.3.3 Specular Reflection	58
3.4	Shading	62
	3.4.1 Gourand Shading or Intensity Interpolation Scheme	63
	3.4.2 Phong Shading or Normal Vector Interpolation Shading	64
3.5	Ray Tracing	69
	3.5.1 Basic Ray Tracing Algorithm	71
3.6	Summary	74
3.7	Solutions/Answers	74

3.1 INTRODUCTION

In unit 2 we had discussed some methods for visible-surface detection, but in order to generate visibility the presence of light is one of the basic requirements. It is obvious that without light and the interaction of light with objects, we would never see anything at all. A study of the properties of light and how light interacts with the surfaces of objects is hence vital in producing realistic images in Computer Graphics. So before considering the production of images to be used in animation or any other application it is worth studying some of the basic properties of light and colour and also to introduce the modeling of the interaction of light with surfaces in Computer Graphics because attainment of realism is one of the basic motives of computer graphics and without considering the effect of light the same cannot be achieved. From Principle Physics we can derive models, called "illumination models", of how light reflects from surfaces and produces what we perceive as color. In general, light leaves some light source, e.g., a lamp or the sun, and is reflected from many surfaces and then finally reflected to our eyes, or through an image plane of a camera. In the overall process of reflection, scattering from the objects in the path of light rays there is always production of shadows and shades with varying levels of intensities; this concept of shading is very important in computer graphics because it also contributes to the realism of the scene under preparation. Ray tracing is one of the exercises performed to attain the realism in a scene. In simple terms Ray Tracing is a global illumination based rendering method used for producing views of a virtual 3-dimensional scene on a computer. Ray tracing is closely allied to, and is an extension of, ray casting, a common hidden-surface removal method. It tries to mimic actual physical effects associated with the propagation of light. Ray tracing handles shadows, multiple Specular reflections, and texture mapping in a very easy straightforward manner. In this unit we have a section dedicated to Ray tracing where we intend to inform you how the basic ray tracing algorithm works. We will take a simple approach for the explanation of the concept, avoiding the mathematical perspective which is traditionally used on the subject. It is intended primarily to inform the curious, rather than to teach the ambitious.

3.2 OBJECTIVES

After going through this unit, you should be able to:

- describe types of light sources and their effects;
- discuss Illumination model and different reflections covered in this model;
- discuss the concept of shading and its types, and
- describe the concept of Ray tracing and algorithms used.

3.3 ILLUMINATION MODEL

Conceptually illumination is exposure of an object to the light, which contributes to light reflected from an object to our eyes and this phenomenon in turn determines the color perceived by an object. Thus, if white light is incident on an object then if that object absorbs green and blue light then we shall perceive it as being red. The colour of the light incident on the surface will determine the colour perceived by the viewer, for **example**, if you see red rose in blue light then it will appear black because all blue rays are absorbed by the object and nothing is reflected so it appears black. Similarly, it is the reflectance of the object surface that determines that an object will appear dull or shining; if the object absorbs a high percentage of the light incident on it then it will appear dull whereas if it reflects a large percentage of the light incident on it then it will appear glossy or shiny. For example, if green light were to shine on a red surface then the surface would be perceived as black because a red surface absorbs green and blue.

Thus, to produce realistic computer-generated images of solid opaque objects the various interactions of light with a surface have to be accounted for, in some form of reflected light and for this the Illumination Model is the gift to Computer Graphics from Physics, which will us help to achieve realism in any graphic scene. An illumination model is also called lighting model and sometimes referred to as shading model, which is used to calculate the intensity of the light that is reflected at a given point on surface of an object. Illumination models can be classified as:

Local illumination model: Where only light that is directly reflected from a light source via a surface to our eyes is considered. No account is taken of any light that is incident on the surface after multiple reflections between other surfaces. This is the type of illumination model that is used in most scan-line rendering pipelines. That is the contribution from the light that goes directly from the light source and is reflected from the surface is called a “local illumination model”. So, for a local illumination model, the shading of any surface is independent of the shading of all other surfaces. The scan-line rendering system uses the local illumination model.

Global illumination model: Global illumination model adds to the local model the light that is reflected from other surfaces to the current surface. A global illumination model is more comprehensive, more physically correct, and produces more realistic images. It is also more computationally expensive. In a Global Illumination Model the reflection of light from a surface is modeled as in the local model with the addition of light incident on the surface after multiple reflections between other surfaces. Although the model is computationally more intensive than a local model but attainment of realism through this model is quite possible. The two major types of graphics systems that use global illumination models are Radiosity and Ray tracing.

Radiosity and Ray tracing (The difference in the simulation is the starting point: Ray tracing follows all rays from the eye of the viewer back to the light sources. Radiosity simulates the diffuse propagation of light starting at the light sources). They produce

more realistic images but are more computationally intensive than scan-line rendering systems which use local illumination model.

Ray tracing: Ray tracing follows all rays from the eye of the viewer back to the light sources. This method is very good at simulating specular reflections and transparency, since the rays that are traced through the scenes can be easily bounced at mirrors and refracted by transparent objects. We will discuss these concepts Reflection/ Refraction/ transparency when we reach the section of ray-tracking.

Radiosity: Radiosity simulates the diffuse propagation of light starting at the light sources. Since global illumination is a very difficult problem and with a standard ray tracing algorithm, this is a very time consuming task, as a huge number of rays have to be shot. For this reason, the radiosity method was invented. The main idea of the method is to store illumination values on the surfaces of the objects, as the light is propagated starting at the light sources.

Deterministic **radiosity algorithms** were used for radiosity for quite some time, but they are too slow for calculating global illumination for very complex scenes. For this reason, stochastic methods were invented, that simulate the photon propagation using a Monte Carlo type algorithm.

Note: **An illumination model** is also called lighting model and sometimes referred to as shading model, which is used to calculate the intensity of the light that is reflected at a given point on the surface of an object, whereas the **Surface rendering algorithm** uses the intensity calculations from an illumination model to determine the light intensity for all projected pixels positions for the various surfaces in the scene.

From the above discussion we have realised that it's the types of light source that contributes a lot towards the attainment of realism in any computer graphics scene.

So, let us discuss the types of light sources. The light sources can not only be natural like light from Sun or Moon or Stars but it could be man-made devices like bulb or tube etc., or a highly polished surface. The light sources are referred as **Luminous objects** which are the objects that emit radiant energy and they can be of both types **light emitting source** (which could be of any type point /diffuse/distributed objects emitting radiant energy) and a **light reflecting source** (Reflecting surfaces are sometimes referred to as light reflecting sources, i.e., any polished surface capable of reflecting, a considerable amount of light rays).

Note: When we view an opaque non-luminous object, we see reflected light from one surface of the object. The total reflected light is the sum of each contribution from light sources and other reflecting surfaces in the scene. Thus a surfaces that is not directly exposed to a light source may still be visible if nearby objects are illuminated.

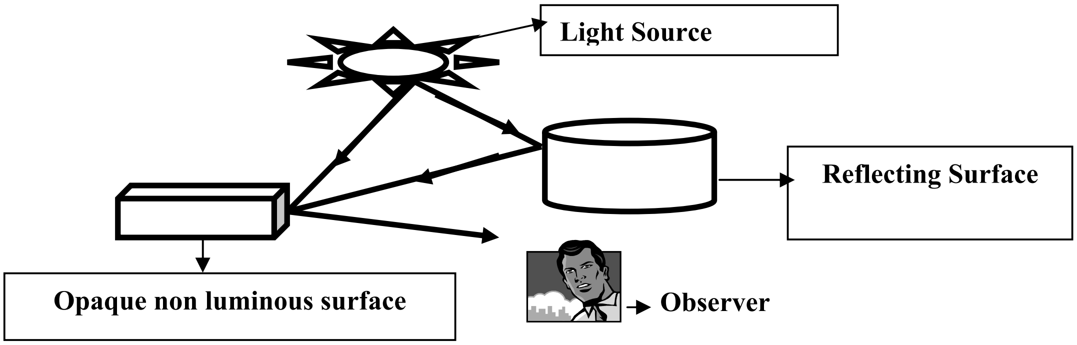


Figure 1

From *Figure 1* we can conclude that the expression given below holds good in real life situations

Light viewed from opaque non-luminous surface = Light from sources + Light from Other Surfaces

Since light sources are quite dominant which are required to establish realism in any graphic scene. Further, there are a variety of light sources, so we need to classify them.

Sources of Light can be classified as:
(a) Point source (b) Parallel Source (c) Distributed Source

Classification of Light Sources

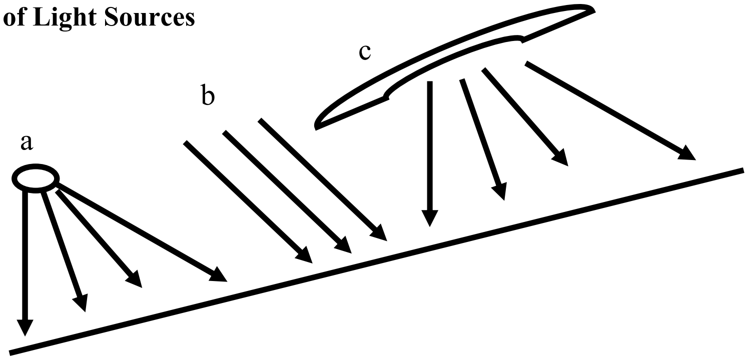


Figure 2: a-Point source; b-Parallel Source; c- Distributed Source

a) **Point source:** It is the simplest model for a light emitter. Here rays from source follow radically diverging paths from the source position, e.g., sources such LED's or small bulbs, i.e., these are the light sources in which light rays originate at a point and radially diverge, such type of sources have dimensions quite smaller as compared to the size of object as shown in *Figure 2*, the source *a* is a point source

b) **Parallel source:** It is to be noted that when point source is at an infinite distance then light rays are parallel and acts as **parallel source** as shown in *Figure 2*, the source b is a parallel source.

c) **Distributed light source:** It models nearby sources such as the long fluorescent light are modeled in category of distributed light source. Here all light rays originate at a finite area in space. Shown in *Figure 2*, the source *c* is a distributed light source

Note: When a light is incident on an opaque surface, part of it is reflected and part of it is absorbed. The amount of incident light reflected by a surface depends on the type of material (shiny material reflect more of the incident light and dull surfaces absorb more of the incident light). Thus, from the reflected amount of light we can deduce many properties of the surface under exposure.

Description of any light source by a luminance, the factors considered are:

Light source described by a luminance

1) Each color (r-red, g-green, b-blue) is described separately

2) $I = [Ir\ Ig\ Ib]^T$ (I for intensity- which is the number of photons incident on a surface in specific time duration).

Now, the interaction of light and the surface under exposure contributes to several optical phenomena like reflection, refraction, scattering, dispersion, diffraction, etc.

☞ Check Your Progress 1

1) Differentiate between Luminous and illuminous objects.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) What will be the colour of a blue rose when it is viewed in red light? Give reasons in support of your answer.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) If the source of light is very far from the object what type of rays you expect from the source? What will happen to the type of rays if source is quite close to the object?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

Let us discuss reflection and its types:

Reflection: It is the phenomenon of bouncing back of light, this phenomenon follows **laws of Reflection** which are:

First Law of Reflection: The Incident ray, the Reflected ray and the Normal all lie on the same plane.

Second Law of Reflection: The angle of Incidence is equal to the angle of Reflection.

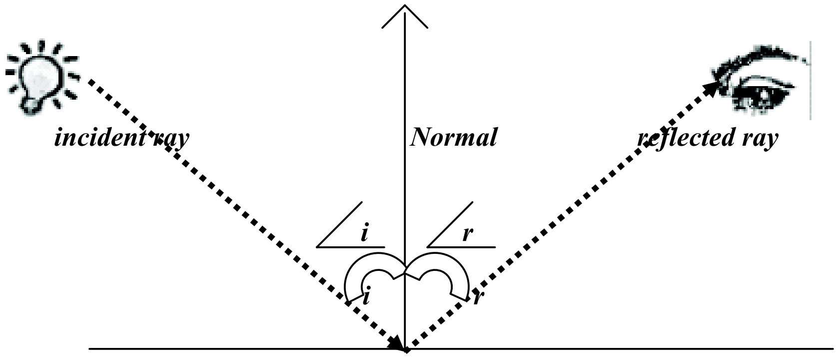


Figure 3: Plane of incidence

Note: Interaction of Light and Surface: A surface has 4 types of interaction with light:

- Diffuse reflection – light is reflected uniformly with no specific direction
- Specular reflection – light is reflected in a specific direction
- Diffuse transmission – light is transmitted uniformly with no preferred direction
- Specular transmission – light is transmitted in a preferred direction.

Types of Reflection: In order to attain realism, this phenomenon of reflection, which occurs due to interaction of light and surface, is needed to be implemented by different ray tracing techniques and other tools. But the usage of tools depends on types of reflection.

- **Ambient Reflection**
- **Diffuse Reflection**
- **Specular Reflection**

Let us discuss different types of reflections.

3.3.1 Ambient Reflection

Whenever we go for the study of light effects, then surroundings play an important role and it is assumed that there exists some light in surroundings falling uniformly on neighbourhood objects. This light in the environment is categorised as Ambient Light (it is non-directional, i.e., it exposes the object uniformly from all directions).

Ambient light is the combination of light reflections from various surfaces to produce uniform illumination which is referred to as Ambient light or Background light. Some features associated with this kind of light are:

- Ambient light has no directional or spatial characteristics,
- The amount of ambient light incident on each object is constant for all surfaces and for all directions.
- The amount of ambient light reflected is dependent on the properties of the surface
- The intensity of ambient light uniform at every point may be different for every surface and color r,g,b

Example: Consider a sphere with a light source above it, thus its lower half will not be illuminated. In practice in a real scene this lower half would be partially illuminated by light that had been reflected from other objects. This effect is approximated in a local illumination model by adding a term to approximate this general light which is 'bouncing' around the scene. This term is called the **ambient reflection** term and is modeled by a constant term. Again the amount of ambient light reflected is dependent on the properties of the surface.It is to be noted that if $I_a \rightarrow$ intensity of ambient light; $K_a \rightarrow$ property of material (**Ambient reflection coefficient k_a, $0 < k_a < 1$**) then resulting reflected light is a constant for each surface independent of viewing direction and spatial orientation of surface.

Say, $I_a \rightarrow$ *Intensity of ambient light.*
$I \rightarrow$ *Intensity of reflected ambient light*
It is assumed that $I_a \neq 0$ ($\because$ $I_a = 0 \Rightarrow$ *These does not exit any light*)
$I \alpha I_a \Rightarrow I = K_a I_a$; $K_a \rightarrow$ *constant* ; $0 \leq K_a \leq 1$
$K_a = 0 \Rightarrow$ *object has absorbed the whole incident light.*
$K_a = 1 \Rightarrow$ *object has reflected the whole incident light.*
$0 \leq K_a \leq 1 \Rightarrow$ *object has reflected some and absorbed some light.*

☞ Check Your Progress 2

1) How does the value of ambient reflection coefficient deduce the property of material?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

2) What should be the K_a for a black hole in universe? Give reasons.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

3.3.2 Diffuse Reflection

Diffuse reflection is characteristic of light reflected from a dull, non-shiny surface. Objects illuminated solely by diffusely reflected light exhibit an equal light intensity from all viewing directions. That is in Diffuse reflection light incident on the surface is reflected equally in all directions and is attenuated by an amount dependent upon the physical properties of the surface. Since light is reflected equally in all directions the perceived illumination of the surface is not dependent on the position of the observer. Diffuse reflection models the light reflecting properties of matt surfaces, i.e., surfaces that are rough or grainy which tend to scatter the reflected light in all directions. This scattered light is called diffuse reflection.

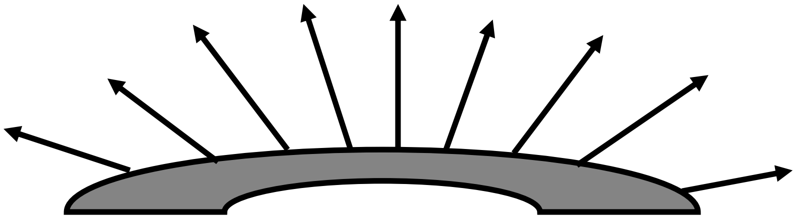


Figure 4: Diffused Reflection From surface

Note:

1) A very rough surface appears equally bright from all viewing directions ∵ the intensity of reflected light is uniform in all directions thus produce diffuse reflection which are constant over each surface in a scene, independent of the viewing direction.

   The fractional amount of the incident light that is diffusely reflected can be set for each surface with parameter K_d . **K_d →diffuse reflection coefficient or diffuse reflectivity.$0 \le K_d \le 1$(K_d → property of material).**

   $K_d = 1$ for highly reflective surfaces reflecting whole light.
   $K_d = 0$ for surfaces that absorb light fully.

2) **Assumption:** i) The diffuse reflections from the surface are scattered with equal intensity in all directions, independent of viewing direction. Such surfaces are called "ideal diffuse reflectors" or "Lambertion reflectors" ∵ radiant high energy from any point on the surface is governed by "LAMBERTS COSINE LAW". (i.e., in diffuse reflection ease the intensity of reflected light (**I**) is α **cos** θ and (ii) K_d ⇔ Ka (generally).

"LAMBERTS COSINE LAW" states that the radiant energy from any small surface area dA in any direction θ relative to the surface normal is proportional to Cos θ.

In case of diffused reflection the source is directional but reflection is uniform.
say,

$I_d \rightarrow$ Intensity of incident diffused light.

Then as per the Lambert's law the intensity of reflected light (I) will be $\alpha \cos\theta$.
Where, θ = Angle between unit direction of incident light vector and unit normal to the surface (or angle of incidence).

/*LAMBERT's LAW * /

$$I = K_d\, I_d \cos\theta$$

$K_d \rightarrow$ diffused reflection coefficient.
$0 \le K_d \le 1$

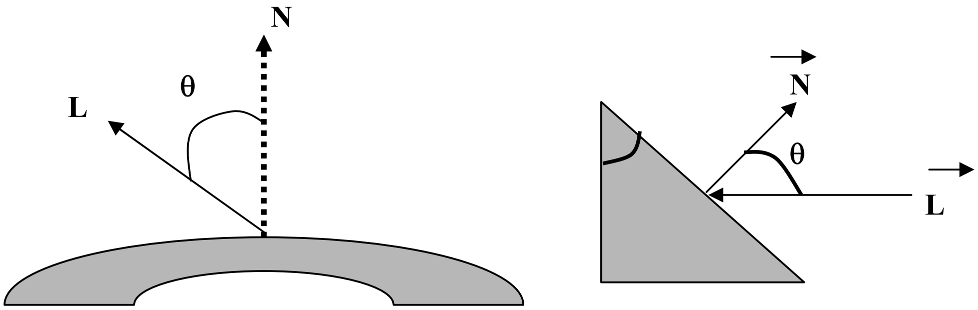


Figure 5

$$I = K_d\, I_d\, (\bar{N}.\bar{L})$$

$I \,\alpha \cos\theta \Rightarrow$ less θ leads to more reflection &
more θ leads to less reflection.

Dot product of N & L vectors $\bar{N}.\,\bar{L} = |\bar{N}|\;|\bar{L}|\,\cos\theta = \cos\theta$ ($\because |\bar{N}|\,\&\,|\bar{L}|$ are) 1" 1"

Unit normal to surface **Unit vector in light direction**

3) **Combined effect of ambient and diffused reflection**

Here the resulting intensity I will be the sum total of the intensities in case 8.3.2 & 8.3.3 we get

$$I = I_a K_a + I_d K_d \cos\theta = I_a K_a + I_d K_d \left(\bar{N}.\bar{L}\right)$$

Take $K_a = K_d$ ($\because$ both constant properties of material to which light is incident, for both sources there constant are same).

Example: Consider a shiny surface with diffused reflection coefficient of 0.8 and ambient reflection coefficient of 0.7, the surface has normal in the direction of 2i + 3j +4k; say some light is incident on it from the direction i + j + k such that the

ambient and diffused intensities are of order 2 and 3 units. Determine the intensity or reflected light.

Solution: The combined effect of ambient and diffused reflection is given by

$$I = I_a K_a + I_d K_d \cos\theta = I_a K_a + I_d K_d (\bar{N}.\bar{L})$$

Using the data given in the equation we get

$$\begin{aligned} I &= 2 * 0.7 + 3 * 0.8 * ((2i + 3j + 4k).(i + j + k)) \\ &= 1.4 + 2.4 (2 + 3 + 4) \\ &= 1.4 + 9*2.4 \\ &= 23 \end{aligned}$$

3.3.3 Specular Reflection

Specular reflection is when the reflection is stronger in one viewing direction, i.e., there is a bright spot, called a specular highlight. This is readily apparent on shiny surfaces. For an ideal reflector, such as a mirror, the angle of incidence equals the angle of specular reflection, as shown below.

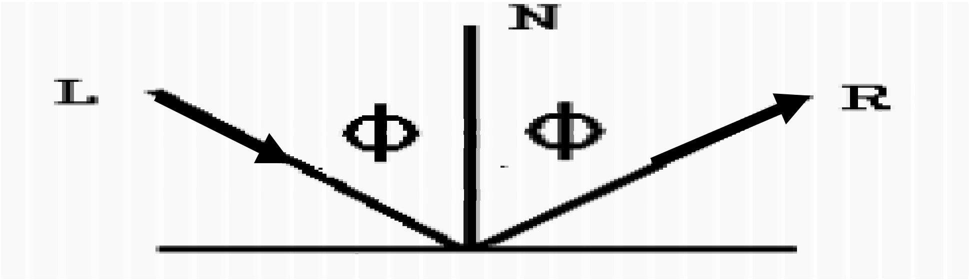


Figure 6

Light is reflected mainly in the direction of the reflected ray and is attenuated by an amount dependent upon the physical properties of the surface. *Since the light reflected from the surface is mainly in the direction of the reflected ray the position of the observer determines the perceived illumination of the surface*. Specular reflection models the light reflecting properties of shiny or mirror-like surfaces.

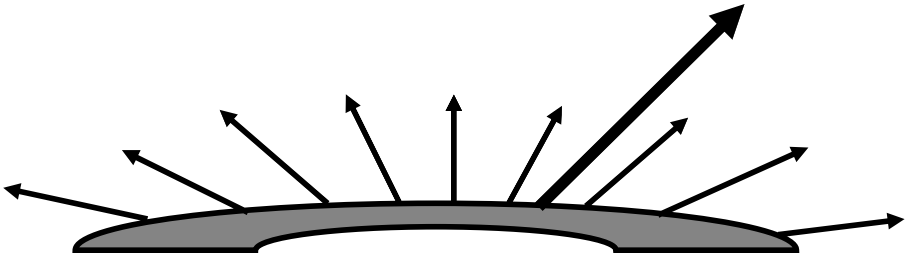


Figure 7

Note: (1) In addition to diffuse reflection, light sources create highlights or bright spots called specular reflection. This highlighting is more pronounced on shiny surfaces than on dull surfaces.

(2) Hence, the local illumination model that is generally used is

illumination = Ambient + Diffuse + Specular

This model of local illumination is usually called the **Phong** specular reflection model.

Let us discuss the concept of specular reflection in a more practical way. Consider the *Figure 9*. Here if **R** is the direction of specular reflection and **V** is the direction of the viewer (located at the View Reference Point or **VRP**), then for an ideal reflector the specular reflection is visible only when **V** and **R** coincide. For real objects (not perfect reflectors) the specular reflectance can be seen even if **V** and **R** don't coincide, i.e., it is visible over range of values (or a cone of values). The shinier the surface, the smaller the f(α) range for specular visibility. So a specular reflectance model must have maximum intensity at **R**, with an intensity which decreases as f(α).

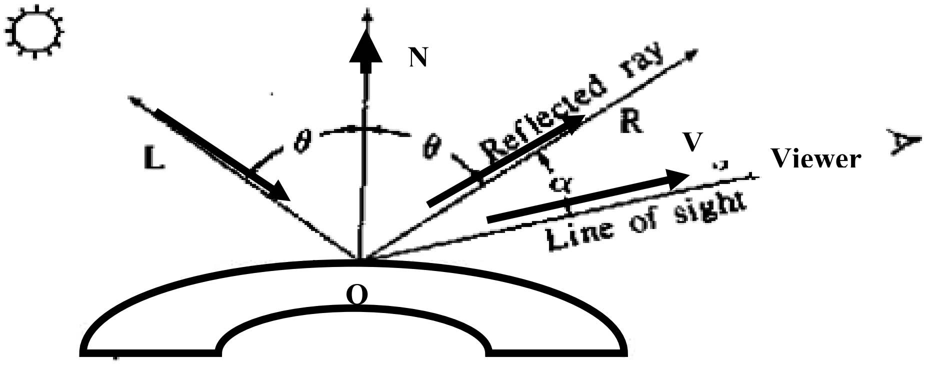


Figure 8

From the above discussion we conclude that Specular reflection is the result of total or near total reflection of the light in a concentrated region around the specular reflection angle (α and the description of other variables shown in *Figure 8* are

$\overline{N}$ → Unit normal surface vector.
$\overline{R}$ → Unit vector in the direction of ideal specular reflection
$\overline{L}$ → Unit vector in the direction of pt. Light source
$\overline{V}$ → Unit vector pointing the viewer.
α → viewing angle relative to $\overline{R}$.

Note:

- At α = 0 viewer will see light of more intensity.
- In case of ideal reflection (perfect mirror) incident light is reflected only in specular reflection direction.
- Objects other than ideal reflection exhibit specular reflection over a finite range of viewing positions around $\overline{R}$ (shiny surfaces have narrow specular reflection range and dull surfaces have wide range).

☞ Check Your Progress 3

1) What will be the change in viewing angle of reflection if the surface under exposure of light is transforming from imperfect reflector to a perfect one?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2) If no variation in the intensity of reflection light is observed in any direction, then what can you say about the smoothness of the surface? Also specify what type of reflection you expect from such surface.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) Discuss the law that forms the basis of Lambertion reflections?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

Phong Model / Phong Specular Reflection Model

This is an empirical model, which is not based on physics, but physical observation. Phong observed that for very shiny surfaces the specular highlight was small and the intensity fell off rapidly, while for duller surfaces it was larger and fell off more slowly. He decided to let the reflected intensity be a function of $(\cos \alpha)^n$ with *n >= 200* for a shiny surface and n small for a dull surface. For a perfect reflector n equals infinity, and for a piece of cardboard *n* equals 0 or 1. In the diagram below we can see how the function $(\cos \alpha)^n$ behaves for different values of *n*. This empirical model for calculating the specular reflection range was developed by Phong and hence called PHONG MODEL/ PHONG SPECULAR REFLECTION MODEL. Which says that, the intensity of specular reflection is proportional to $\cos^n\alpha$ (α lies between 0° & 90°) so $\cos \alpha$ varies from 1 to 0).where '*n*' is *specular reflection parameter* dependent on the type of surface.

Notice that the Phong illumination equation is simply the Lambert illumination equation with an additional summand to account for specular reflection and ambient reflection.

Intensity of specular reflection depends on material properties of surface and the angle of incidence and the value of *specular reflection parameter* '**n**' is determined by the type of surface, we want to display.

- shiny surfaces are modeled with larger values of *n* (100 or more (say))
- dull surfaces are modeled with smaller values of *n* (down to 1)
- for perfect reflection *n* is ∞.
- for very dull surfaces (eq chalk etc) *n* is assigned value near to 1.
- In the diagram below we can see how the function $(\cos \alpha)^n$ behaves for different values of n.

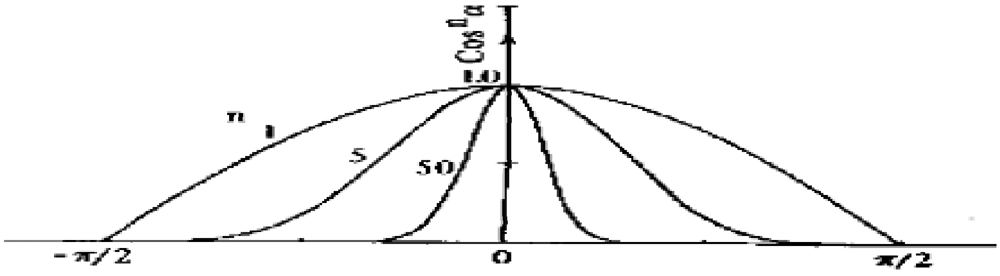


Figure 9: The plot of $\cos^n \alpha$ with α

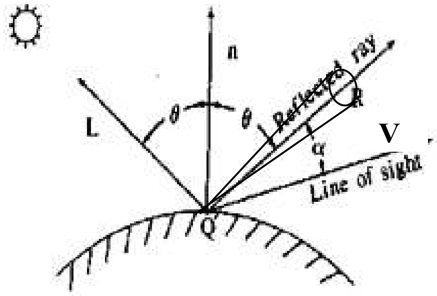


Figure 10: Shiny surface(large n)

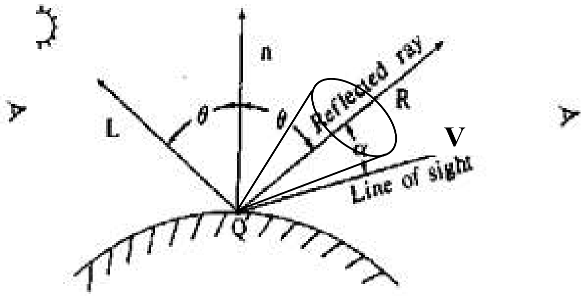
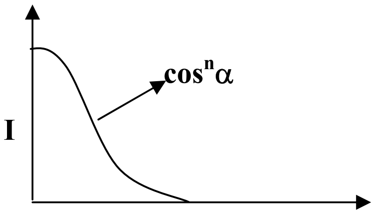
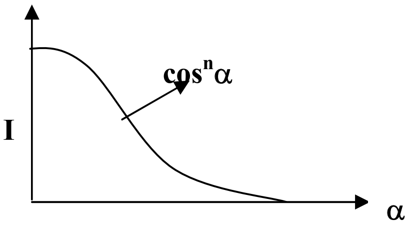


Figure 11: Dull surface(Small n)

As per the phong model variation of Intensity (I) with α (because I α $\cos^n \alpha$) is



i) for shiny surface (n > 1)



ii) Dull surface (n ~= 1)

As per the Phong specular reflection,

$$I \text{ or } I_{spec.} = I_s K_s \cos^n \alpha$$

where, $I_s \rightarrow$ intensity of source
I OR $I_{spec} \rightarrow$ intensity of specular reflected light
$K_s \rightarrow$ Specular reflection coefficient

resulting intensity in the case when all ambient/diffuse/ specular reflection occurs is

$$I = I_a K_a + I_d K_d \cos\theta + I_s K_s \cos^n \alpha$$

Now, $\cos^n \alpha = (\bar{R}.\bar{V})^n$ $\quad (\because \cos\alpha = \bar{R}.\bar{V})$.

Where,
$\bar{R} \rightarrow$ Unit vector in specular reflection direction
$\bar{V} \rightarrow$ Unit vector pointing the viewer

i.e. $|\bar{V}| = 1$; $|\bar{R}| = 1$.
$\bar{R}.\bar{V} = |\bar{R}|\ |\bar{V}| \cos\alpha = 1.1. \cos\alpha = \cos\alpha.$
($\Rightarrow \cos\alpha = \bar{R} . \bar{V}$)

Example: *Calculate* $\bar{R}.\bar{V}$ *using* $\bar{N}, \bar{L}$ & $\bar{V}$ *where the variables have their respective meanings* $\bar{N}$ $\rightarrow$ Unit normal surface vector.
$\bar{R}$ $\rightarrow$ Unit vector in the direction of ideal specular reflection
$\bar{L}$ $\rightarrow$ Unit vector in the direction of pt. Light source
$\bar{V}$ $\rightarrow$ Unit vector pointing the viewer.

Solution: Consider the figure given below

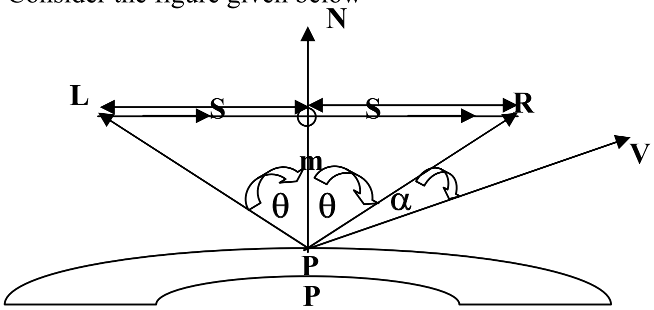


Figure 12

$|\overline{Pm}| = |\overline{L}\cos\theta| = \cos\theta$ ------------------(1) $\qquad (|\overline{L}| = 1 \quad (\because \overline{L}$ is unit vector))

$\overline{Pm} = |\overline{Pm}|\,.\ \hat{Pm} = |\overline{Pm}|\ \overline{N}$ ($\hat{Pm}$ has direction of normal $\overline{N}$)

$\Rightarrow \overline{Pm} = \overline{N}\cos\theta$ ----------(2)

By figure vectors S and R can be found to be

$\overline{S} = \overline{Pm} - \overline{L}$ ----------------------------- (3)

$\overline{R} = \overline{Pm} + \overline{S}$ -----------------------------(4)

Use (3) in (4) we get

$\overline{R} = 2\,\overline{Pm} - \overline{L}$ --------------------------------(5)

Use (2) in (5) we get

$\overline{R} = 2\,(\overline{N}\cos\theta) - \overline{L}$

$\Rightarrow \overline{R}\,.\,\overline{V} = [\,2\,(\overline{N}(\cos\theta) - \overline{L}\,]\,.\,\overline{V}\,.$

$\overline{R}\,.\,\overline{V} = [2\,(\overline{N}(\overline{N}\,.\,\overline{L}) - \overline{L}\,]\,.\,\overline{V}$ $\qquad (\because \cos\theta = \overline{N}\,.\,\overline{L}).$

3.4 SHADING

When an object is under the exposure of light, then the rays of light are distributed over the surface and the distribution of intensity pattern depends very much on the shape of object. Now to represent such 3D scene on computer we need to do rendering, i.e., transforming the 3D image into a 2D image, and because of this rendering there is a possibility of loss of information like depth, height, etc., of an object in the scene. So to preserve this takes different Illumination models into consideration, for preserving the information embedded in 3D scene & let it not be lost while transforming it in to 2D scene.

So let us study in some detail the type of shading and rendering techniques.

Polygon –Rendering Methods

Here we will consider the application of an illumination model to perform the rendering of standard graphics objects, i.e., those formed with polygonal surfaces. Scan line algorithm typically applies lighting model to perform rendering of polygon surfaces, by using any one of the two ways:

1) Each polygon can be rendered with a single intensity.
2) The intensity can be obtained at each point of the surface using an interpolation scheme.

The result of two ways leads to 3 types of shading:

(a) **Constant intensity shading OR Flat shading**

In this method single intensity is calculated for each polygon surface i.e., all points which lie on the surface of the polygon are displayed with the same intensity value. This constant shading is useful for quickly displaying the general appearance of a curved surface, but this shading does not convey explicit information about the curved surface.

(b) **Gourand shading OR Intensity interpolation scheme**
We will discuss this scheme in successive section 3.4.1.

(c) **Phong shading OR Normal vector interpolation shading.**
We will discuss this scheme in successive section 3.4.2.

3.4.1 Gourand shading OR Intensity interpolation scheme

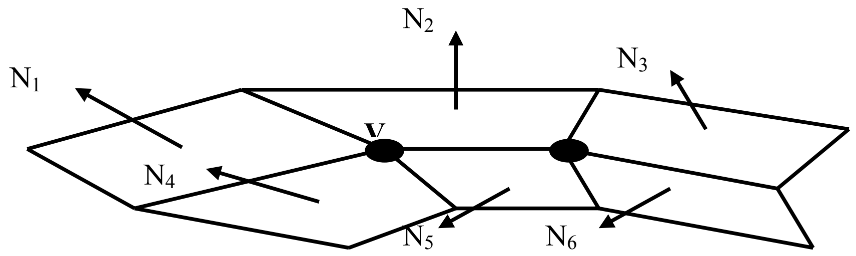


Figure 13

Here polygon is rendered by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Calculations to be performed for each polygon surface rendered with Gourand shading:

1) Determine average unit normal vector at each polygon vertex.
2) Apply illumination model to each vertex to calculate the vertex intensity.
3) Linearly interpolate the vertex intensities over the surface of the polygon.

i) *To determine average unit normal vector at each polygon vertex:*

At each polygon vertex (as shown by point V in the figure), the normal vector is obtained by averaging the surface normal of all polygons sharing that vertex. Thus, for any vertex *V* the unit vertex normal will be given by $\overrightarrow{N_v}$

$$\overline{N_V} = \frac{\sum_{=1}^{n} N_k}{|\sum_{k=1} N_k|}$$

$K \rightarrow 1$ to n are the surfaces in contact with the vertex *v*.

ii) *How to use illumination model to calculate vertex intensity:*

For this we interpolate intensities along the polygon edges, for each scan line the intensity at the intersection of the scan line with a polygon edge is linearly interpolated from intensities at the edge end points, i.e., by using parametric equation of line discussed in Unit 2 we can find intensity at point 4, i.e., I_4; during the evaluation of I_4 intensities at point. 1 & 2 i.e., I_1 & I_2 are used as the extreme intensities. So I_4 is linearly interpolated from intensities at the edge end points I_1 & I_2 (Refer to the Figure given below).

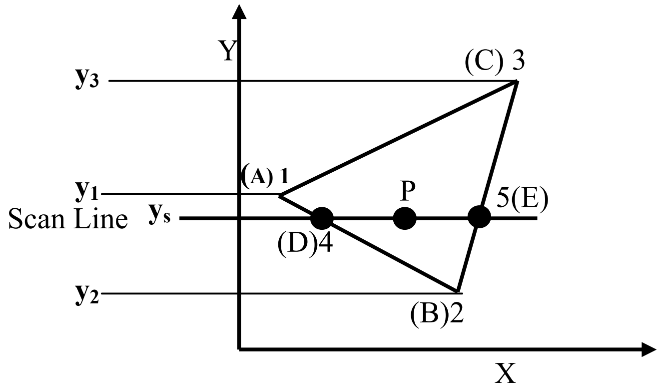


Figure 14

(iii) Linearly interpolate

$$
\begin{cases}
I_4 = I_1 + t(I_2 - I_1) & \text{where } t = \dfrac{|y_1 - y_s|}{|y_1 - y_2|} \\
I_D = I_A + t(I_B - I_A) & \text{where, } t = \dfrac{|AD|}{|AB|} \\
\text{similarly } I_E = I_C + t(I_B - I_C) & \text{where } t = \dfrac{|CE|}{|CB|} \\
\& \, I_P = I_D + t\ (I_E - I_D) & \text{where } t = \dfrac{|EP|}{|DE|}
\end{cases}
$$

where I_p → Intensity of points over the surface of polygon i.e., in Gourand shading the intensity of point - 4 is linearly interpolated from intensity at vertices 1 and 2, similarly of point 5 too is interpolated from intensity at vertices 3 and 2. Intensity of points *P* is linearly interpolated from intensity at point 4 and 5.

Advantages of Gourand Shading: It removes the intensity discontinuities associated with the constant shading model.

Deficiencies: Linear intensity interpolation can cause bright and dark streaks called Mach bands to appear on the surface, these mach bands can be removed by using Phong shading or by dividing the surface into greater number of polygon faces.

Note: In Gourand Shading because of the consideration of average normal, the intensity is uniform across the edge between two vertices.

3.4.2 Phong shading OR Normal Vector Interpolation Shading

In Gouraud shading we were doing direct interpolation of intensities but a more accurate method for rendering a polygon surface is to interpolate normal vectors and then apply illumination model to each surface. This accurate method was given by Phong and it leads to Phong shading on Normal vector interpolation shading.

Calculations involved with Phong Shading:

i) Determine average unit normal vector at each polygon vertex.
ii) Linearly interpolate the vertex normals over the surface of polygon.
iii) Apply illumination model along each scan line to calculate projected pixel intensities for surface points.

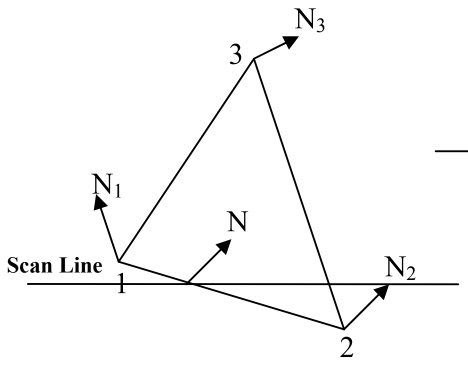


Figure 15

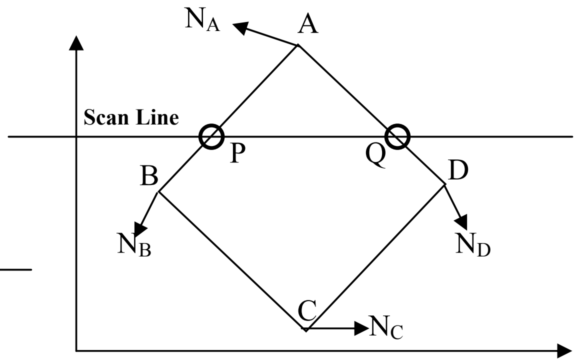


Figure 16

Interpolation of surface normals along the polygonedge between two vertices is shown above in *Figure 15*. The normal vector *N* for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge end points normals. Then incremental methods are used to evaluate normals between scan lines and along each individual scan line. At each pixel position along a scan line , the illumination model is applied to determine the surface intensity at that point

$$N= [((y-y_2) / (y_1-y_2)) N_1] + [((y-y_2) / (y_1-y_2)) N_2]$$

In *Figure 15* above, say, $\overline{N}$ is surface normal to be interpolated along polygon edge 1– 2 having vertices 1 & 2. Such that $\overline{N_1}$ & $\overline{N_2}$ are normal at the vertices. Thus, by using the parametric equation across the edge 1 – 2 we can determine the value of the normal *N*, which will be given by

$$\overline{N} = \overline{N_1} + t\,(\overline{N_2} - \overline{N_1})$$

Similarly, in *Figure 16* we can find $\overline{N}_p$ and $\overline{N}_q$ which are the normal at point (*P* and *Q*) through which the scan line passes,

$$\boxed{\overline{N_P} = \overline{N_A} + t\,(\overline{N_B} - \overline{N_A})} \quad \text{where, } t = \frac{|AP|}{|AB|}$$

Now we use $\overline{N_P}$ to find $\cos\theta$ where , θ is the angle between Normal vector and direction of light represented by vector L(refer to Phong model).

$$\cos\theta = \overline{N}.\,\overline{L} \quad \text{and} \quad \cos^n\alpha = (\overline{R}.\overline{V})^n = [(2\overline{N}(\overline{N}.\overline{L}) - \overline{L}).\ V]^n$$

Now using $\cos^n\alpha$, $\cos\theta$ in

$$\boxed{I = I_a K_a + I_d K_d \cos\theta + I_s K_s \cos^n\alpha}$$

/* similarly we can find intensity of points lying inside the surface */

This N_p will be used to find intensity value i.e., I_P. at points P_o in the object whose projection is P, by using the intensity calculation formula which we had used for the determination of intensities in diffused and specular reflection.

Merit

So by finding intensities at different points across the edge we find that intensity is varying across the edge between two vertex points and is not uniform as in Gouraud Shading, giving much better effect. So we can say that the normal vector interpolation (Phong shading) is superior to intensity interpolation technique (Gourand Shading) because it greatly reduces the mach bands.

Demerit: Requires lot of calculations to find intensity at a point, thus increases the cost of shading in any implimentation.

Problem with Interpolated Shading

There are some more shading models which intermediate in complexity between Gouraud and Phong shading, involving the liner interpolation of the dot products used in the illumination models. As in Phong shading, the illumination model is evaluated at each pixel, but the interpolated dot products are used to avoid the expense of computing and normalizing any of the direction vectors. This model can produce more satisfactory effects than Gouraud shading when used with specular-reflection illumination models, since the specular term is calculated separately and has power-law, rather than linear, falloff. As in Gouraud shading, however, highlights are missed if they do not fall at a vertex, since no intensity value computed for a set of interpolated dot products can exceed those computed for the set of dot products at either end of the span.

There are many problems common to all these interpolated-shading models, several of which we listed here.

Polygonal silhouette: No matter how good an approximation an interpolated shading model offers to the actual shading of a curved surface, the silhouette edge of the mesh is still clearly polygonal. We can improve this situation by breaking the surface into a greater number of smaller polygons, but at a corresponding increase in expense.

Perspective distortion. Anomalies are introduced because interpolation is performed after perspective transformation in the 3D screen-coordinate system, rather than in the WC system. For example, linear interpolation causes the shading information to be incremented by a constant amount from one scan line to another along each edge. Consider what happens when vertex 1 is more distant than vertex 2. Perspective foreshortening means that the difference from one scan line to another in the untransformed *z* value along an edge increases in the direction of the farther coordinate. Thus, if $y_s = (y_1 + y_2)$, then $I_s = (I_1 + I_2)/2$, but z_s will not equal $(z_1 + z_2/2$. This problem can also be reduced by using a larger number of smaller polygons. Decreasing the size of the polygons increases the number of points at which the information to be interpolated is sampled, and therefore increases the accuracy of the shading.

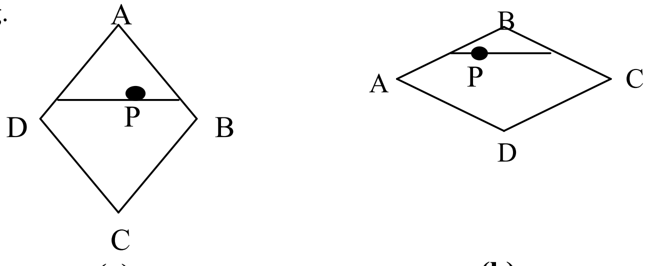


Figure 17: (a) **(b)**

Figure 17 (a) and 17 (b) shows Interpolated values derived for point *P* on the same polygon at different orientations which differ from (a) to (b). *P* interpolates A, B, D in (a) and A, B, C in (b).

Orientation dependence: The results of interpolated-shading models are not independent of the projected polygon's orientation. Since values are interpolated between vertices and across horizontal scan lines, the results may differ when the polygon is rotated (see Figure 17). This effect is particularly obvious when the orientation changes slowly between successive frames of an animation. A similar problem cab also occurs in visible-surface determination when the z value at each point is interpolated from the z values assigned to each vertex. Both problems can be solved by decomposing polygons into triangles. Alternatively, the solution is rotation-independent, but expensive, interpolation methods that solve problem without the need for decomposition.

Problems at shared vertices: Shading discontinuities can occur when two adjacent polygons fail to share a vertex that lies along their common edge. Consider the three polygons of *Figure 17*, in which vertex *C* is shared by the two polygons on the right, but not by the large polygon on the left. The shading information determined directly at *C* for the polygons at the right will typically not be the same as the information interpolated at that point from the values at *A* and *B* for the polygon at the left. As a result, there will be a discontinuity in the shading. The discontinuity can be eliminated by inserting in the polygon on the left an extra vertex that shares *C's* shading information. We can preprocess a static polygonal database in order to eliminate this problem; alternatively, if polygons will be split on the fly (e.g., using the BSP-tree visible-surface algorithm), then extra bookkeeping can be done to introduce a new vertex in an edge that shares an edge that is split.

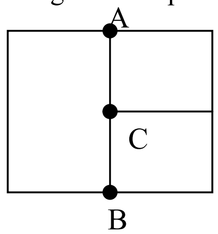


Figure 18: Vertex C is shared by the two polygons on the right, but not by the larger Rectangular polygon on the left.

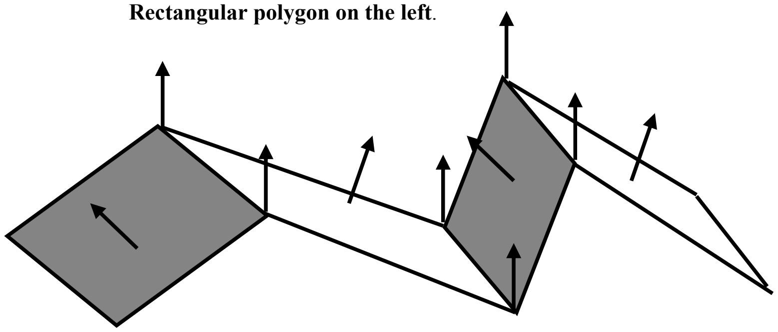


Figure 19: Problems with computing vertex normal. Vertex normal are all parallel.

Unrepresentative vertex normals: Computed vertex normals may not adequately represent the surface's geometry. For example, if we compute vertex normals by averaging the normals of the surfaces sharing a vertex, all of the vertex normals of *Figure 19* will be parallel to one another, resulting in little or no variation in shade if the light source is distant. Subdividing the polygons further before vertex normal computation will solve this problem.

Although these problems have prompted much work on rendering algorithms that handle curved surfaces directly, polygons are sufficiently faster (and easier) to process that they still form the core of most rendering systems.

☞ Check Your Progress 4

1) If specular reflection parameter is small, say 50, than what can you say about the nature of surface? How the nature of surface will change if n starts increasing?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

2) Usage of linear interpolation scheme to obtain intensity at each polygon of surface leads to which type of shading?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

3) What are merits & demerits of Ground shading and Phong shading?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

4) Which shading scheme is best constant shading, Ground shading or Phong Shading? Give reasons to support your answer.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

5) Compare Constant shading, Gourand shading and Phong Shading.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

6) Why Phong Shading is better than Gourand shading.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

7) How Ambient, Diffused and Specular reflection contributes to the resulting intensity of reflected ray of light? Give mathematical expression for the same.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

8) What do you mean by polygon rendering?

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

3.5 RAY TRACING

Ray tracing is an exercise performed to attain the realism in a scene. In simple terms Ray Tracing is a global illumination based rendering method used for producing views of a virtual 3-dimensional scene on a computer. Ray tracing is closely allied to, and is an extension of, ray-casting, a common hidden-surface removal method. It tries to mimic actual physical effects associated with the propagation of light. Ray tracing handles shadows, multiple specular reflections, and texture mapping in a very easy straight-forward manner. So, the **crux** is "Ray tracing is a method of generating realistic images by computer, in which the paths of individual rays of light are followed from the viewer to their points of origin". Any program that implements this method of ray tracing is ray tracer. One of the prime **advantages** of method of Ray tracing is, it makes use of the actual physics and mathematics behind light. Thus the images produced can be strikingly, life-like, or "photo-realistic".

In this section we will discuss the basic ray-tracing algorithm. It will also describe the concept behind anti-aliasing, a method for improving the realism of an image by smoothing the jagged edges caused by the digital nature of computer displays. This section will not involve any discussions of the more advanced features of today's ray-tracers, such as motion blur, depth of field, penumbras (soft shadows), texture mapping, or radiosity.

So to proceed the journey of ray tracing we will begin with basics like concept of **scene** and describe its basic elements. With this as a foundation, it will then introduce **ray casting**, and then **ray tracing** as an extension of ray casting. Finally, the section will discuss the basic concepts behind **anti-aliasing** as a means of improving the realism of an image, and will then conclude with an overview of **how and where ray tracing is used.**

Scenes

In the context of ray tracing, a scene is a collection of objects and light sources that will be viewed via a camera. Each of these items are arranged in what is called the world, or world space which is an imaginary place with height, width and depth (much like reality). For instance, let's suppose that you wish to create an image of the Planets and their satellites. Then our scene will consist of the planets, their respective satellites, a light source that will act as the sun, and our camera, which is where we are viewing this scene from. Let us discuss some basic terms (objects, light source, world, etc.) involved with the concept of scene in some detail.

Objects could be any state of matter (solid, liquid. gas, plasma). Although ray tracers can only support objects that can have mathematical description (such as cube, spheres, cylinders, planes, cones, etc.), various combinations of these basic objects help in creating more complex objects.

It is important to note that all objects have some kind of texture, which includes the color of the object, as well as any bumpiness, shininess, or design that the designer of the image may wish to use. However, to simplify the discussion of how ray tracing works, we will consider color to be the only texture present on the objects that will be described.

Light Sources are key elements in any ray traced scene, because without them, there would be no rays to trace. Light sources are like objects which may be placed at arbitrary locations in the scene, but in addition to the location a light source has some intensity associated with it, which describes the brightness and color of the light. At

any rate, lighting is considered by many to be one of the most important factors in a ray traced image. It is the location and intensity of light source which give liveliness to an image, A picture flooded with too much light might lose any sense of mystery you desired it to have, whereas an image that is too dark will not show enough detail to keep its viewers interested.

Camera represents "eye" or "viewpoint" of observer. In order to describe the basic working of camera we can refer to the working of "pin-hole camera" as shown in the figure below:

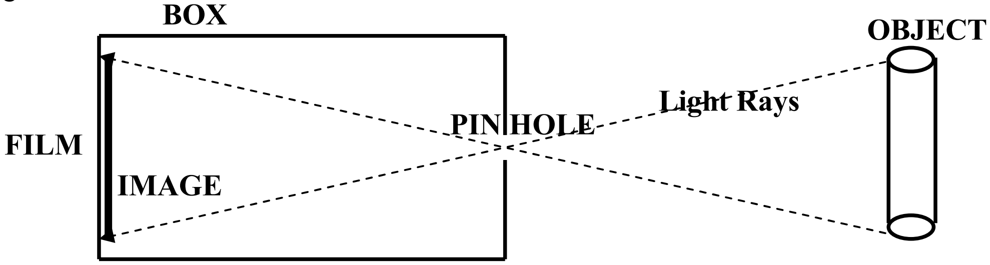


Figure 20

The hole (or "aperture") must be so small that it must prevent light from saturating (and thus overexposing) the film. It allows only a little bit of light into the box at a time. This kind of camera, though simple, is quite effective. It works because light from a given position on the object may come from only one direction and strike only one position on the film. If the hole were any larger, the image would become blurrier as a result of the increased amount of light hitting each spot on the film

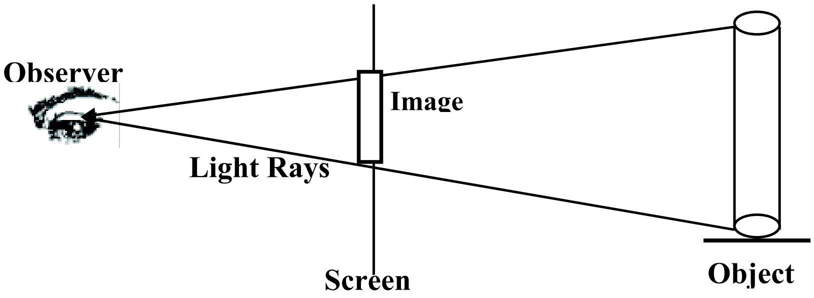


Figure 21

In ray tracing, the camera is much like this, in that it determines where on the "film" (or, in the case of ray tracing, the computer screen) the light rays hit such that a clear realistic rendered image is available. In order to have a better understanding of the topic let us have a discussion on the concept of ray casting also.

Ray Casting

Ray casting is a method in which the surfaces of objects visible to the camera are found by throwing (or casting) rays of light from the viewer into the scene. The idea behind ray casting is to shoot rays from the eye, one per pixel, and find the closest object blocking the path of that ray – think of an image as a screen-door, with each square in the screen being a pixel. This is then the object the eye normally sees through that pixel. Using the material properties and the effect of the lights in the scene, this algorithm can determine the shading of this object. The simplifying assumption is made that if a surface faces a light, the light will reach that surface and not be blocked or in shadow. The shading of the surface is computed using traditional 3D computer graphics shading models. Ray casting is not a synonym for ray tracing, but can be thought of as an abridged, and significantly faster, version of the ray tracing algorithm. Both are image order algorithms used in computer graphics to render three dimensional scenes to two dimensional screens by following rays of light

from the eye of the observer to a light source. Although ray tracing is similar to ray casting, it may be better thought of as an extension of ray casting we will discuss this in the next topic under this section.

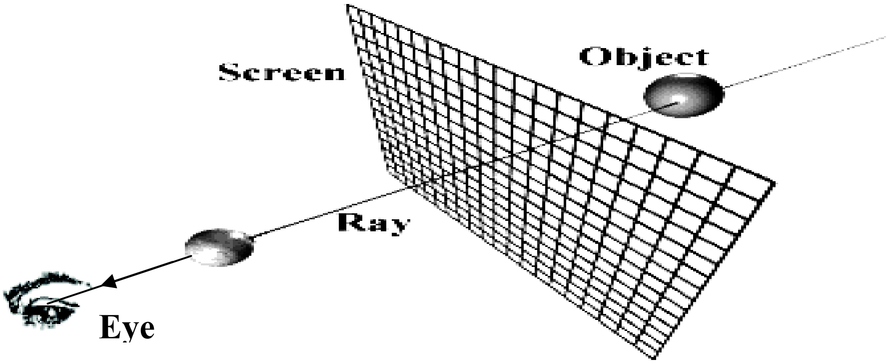


Figure 22

Ray Tracing

"Ray tracing" is a method of following the light from the eye to the light source. Whereas ray casting only concerns itself with finding the visible surfaces of objects, ray tracing takes that a few steps further and actually tries to determine what each visible surface looks like. Although it will cost your processor time spent in calculations you can understand the level of calculations involved in ray tracing by considering this example,

Let's say we are rendering (that is, ray tracing) a scene at a resolution of 320 pixels wide by 240 pixels high, for a total of 76,800 pixels. Let it be of low complexity, with only 20 objects. That means, over the course of creating this picture, the ray tracer will have done 20 intersection tests for each of those 76,800 pixels, for a total of 1,536,000 intersection tests! In fact, most ray tracers spend most of their time calculating these intersections of rays with objects, anywhere from 75 to 95 % of a ray tracer's time is spent with such calculations. Apart from such hectic calculations, there is the good news that there are ways to decrease the number of intersection tests per ray, as well as increase the speed of each intersection test. In addition to this the bad news is that ray tracing complicates things much more than simply ray casting does.

Ray tracing allows you to create several kinds of effects that are very difficult or even impossible to do with other methods. These effects include three items common to every ray tracer: reflection, transparency, and shadows. In the following paragraphs, we will discuss how these effects fit naturally into Ray tracing.

3.5.1 Basic Ray Tracing Algorithm

The Hidden-surface removal is the most complete and most versatile method for display of objects in a realistic fashion. The concept is simply to take one ray at a time, emanating from the viewer's eye (in perspective projection) or from the bundle of parallel lines of sight (in parallel projection) and reaching out to each and every pixel in the viewport, using the laws of optics.

Generally, to avoid wastage of effort by rays starting from sources of light going out of the viewport, the reverse procedure of starting with ray from the viewpoint and traveling to each pixel in the viewport is adopted.

If the ray encounters one or more objects, the algorithm filters out all but the nearest object, or when there is an overlap or a hole, the nearest visible portion of all the objects along the line of sight.

Depending on the nature (attributes) of the surface specified by the user, the following effects are implemented, according to rules of optics.

a) Reflection (according to the angle of incidence and reflectivity of the surface).

b) Refraction (according to the angle of incidence and refraction index).

c) Display of renderings (texture or pattern as specified), or shadows (on the next nearest object or background) involving transparency or opacity, as the case may be.

Figure illustrates some of the general principles of ray tracing.

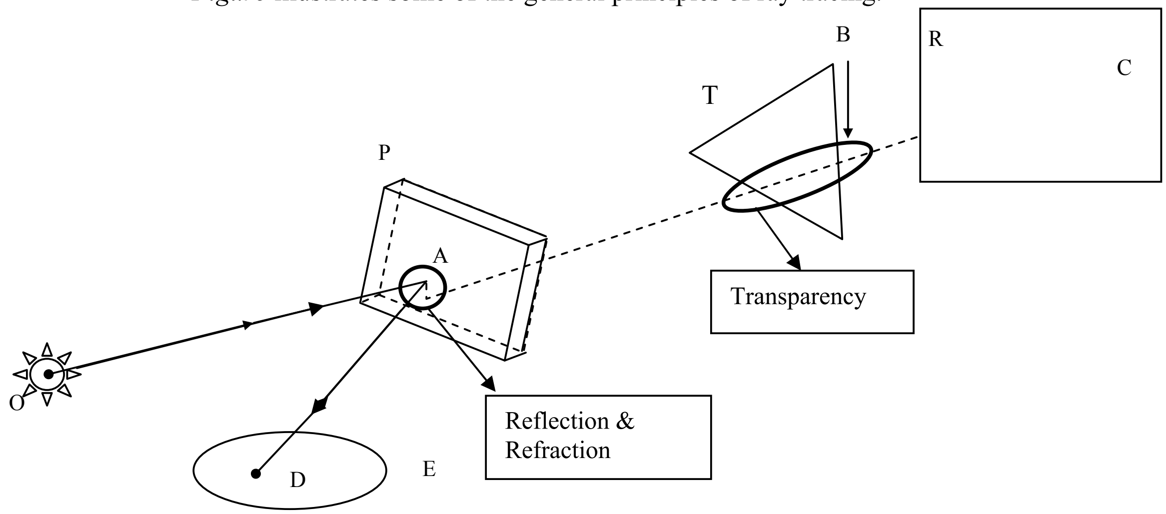


Figure 23

A ray starting at *O* hits the transparent glass plate *P* at an angle at *A*. It gets refracted in the plate (indicated by the kink within the plate thickness). The exiting ray happens to hit the edge of the triangle *T*, and casts a shadow on the opaque rectangular plate *R* at the point *C*. A part of the ray incident on the plate *P* gets reflected at *A* and the reflected ray hits the elliptical object *E* at the point *D*. If *P* is a green glass plate, the exiting ray *AC* will be assigned the appropriate green colour. If *R* or *E* has a textured surface, the corresponding point *C* or *D* will be given the attributes of the surface rendering.

If *O* is a point source of light, the ray *OC* will locate the shadow of the point *B* on the edge of the triangle *T* at the point *C* on the rectangle *R*.

Different locations of light sources may be combined with differing view positions to improve the realism of the scene. The method is general also in the sense that it can apply to curved surfaces as well as to solids made of flat polygonal segments. Because of its versatile and broad applicability, it is a "brute force" method, involving massive computer resources and tremendous computer effort.

Algorithm

Often, the basic ray tracing algorithm is called a "recursive" (obtaining a result in which a given process repeats itself an arbitrary number of times) algorithm. *Infinite recursion* is recursion that never ends. The ray tracing algorithm, too, is recursive, but it is finitely recursive. This is important, because otherwise you would start an image rendering and it would never finish!

The algorithm begins, as in ray casting, by shooting a ray from the eye and through the screen, determining all the objects that intersect the ray, and finding the nearest of those intersections. It then *recurses* (or repeats itself) by shooting more rays from the point of intersection to see what objects are reflected at that point, what objects may be seen through the object at that point, which light sources are directly visible from that point, and so on. These additional rays are often called secondary rays to differentiate them from the original, primary ray. As an analysis of the above discussion we can say that we pay for the increased features of ray tracing by a dramatic increase in time spent with calculations of point of intersections with both the primary rays (as in ray casting) and each secondary and shadow ray. Thus achieving good picture quality, is not an easy task, and it only gets more expensive as you try to achieve more realism in your image. One more concept known as Anti-aliasing is yet to be discussed, which plays a dominant role in achieving the goal of realism.

Anti-aliasing

Anti-aliasing is a method for improving the realism of an image by removing the jagged edges from it. These jagged edges, or "jaggies", appear because a computer monitor has square pixels, and these square pixels are inadequate for displaying lines or curves that are not parallel to the pixels and other reason is low sampling rate of the image information, which in turn leads to these jaggies (quite similar to star casing discussed in previous blocks under DDA algorithm). For better understanding, take the following image of darkened circle:

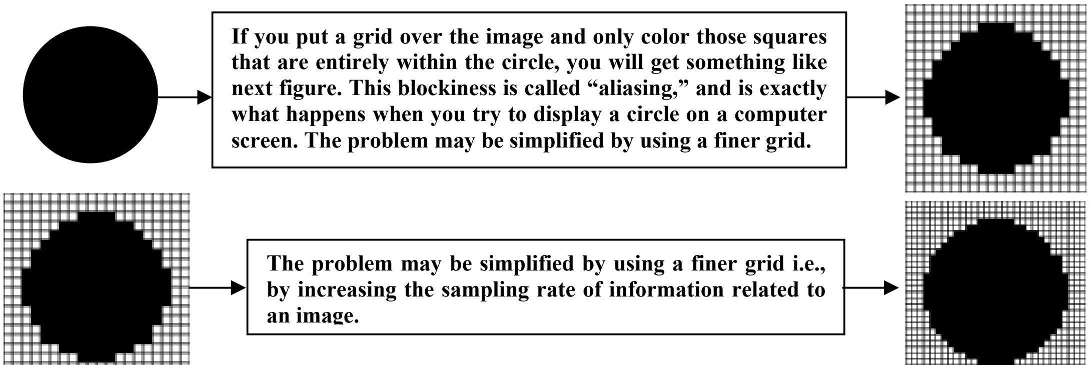
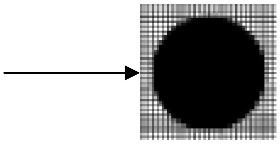


Figure 24

It is not possible to completely eliminate aliasing because computers are digital (discrete) in nature. However, it is possible to minimize aliasing, the solutions used by ray tracers today involve treating each pixel as a finite square area (which, in fact, they are), rather than as a mere point on the screen. Instead the pixel should not be considered as a point or area but should be considered as a sample of image information (higher the sampling is lesser the aliasing is). Now let us discuss how appropriately the sampling can be done - Rays are fired into the scene through the centers of the pixels, and the intensities of adjacent rays are compared. If they differ by some pre-determined amount, more rays are fired into the surfaces of the pixels. The intensities of all the rays shot into a given pixel are then averaged to find a color that better fits what would be expected at that point.

Note: Do not treat a pixel as a square area, as this does not produce correct filtering behaviour, in fact a pixel is not a point, but it is a sample of information to be displayed.

Anti-aliasing, then, helps eliminate jagged edges and to make an image seem more realistic.Continuing the above example, the anti-aliased circle might, then, be represented



Applications of Ray Tracing

So, you might ask, just what practical uses does ray tracing have

- "simulation of real-world phenomena for vision research,
- medical (radiation treatment planning),
- seismic (density calculations along a ray),
- mechanical engineering (interference checking),
- plant design (pipeline interference checking),
- hit-testing in geometric applications, and impact and penetration studies".
- widely used in entertainment.

This topic discussed has just scratched the surface of the topic of ray tracing. In addition to the primitive features of reflection, transparency, and shadows, most ray tracers today support different texture mapping options, focal blur, radiosity, motion blur, and a host of other advanced features. But they are out of the scope of this unit.

☞ **Check Your Progress 5**

1) Differentiate between Ray tracing and Ray Casting.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) How Ray does tracing contribute to achieving realism in computer graphics?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) What is the problem of Aliasing? How does the technique of anti-aliasing work to get rid of the problem of aliasing?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3.6 SUMMARY

In the unit, we have discussed various sources of light and the type of reflections produced by the object under exposure of these sources. When an object is exposed to light various phenomena occur such as reflection, refraction, etc. Out of them shading is most important as the same helps in achieving the realism in computer graphics. We have also discussed the concept of polygon rendering, and ray tracing which are useful in achieving photo realism of a 3D scene.

3.7 SOLUTIONS / ANSWERS

Check Your Progress 1

1) Luminous objects are independent sources of light e.g., Sun, whereas illuminous objects are those sources of light which depend on luminous sources to produce their half of light, e.g., Moon.

2) It will appear black because nothing of the incident light is reflected back.

3) When the source of light is far off, i.e., at infinite distance (comparatively very large distance) the rays of light become parallel, hence the source behaves as a parallel source.

   But as the source of light appears closer to the object these rays of light start following radially outward path and hence transform to point source of light.

Check Your Progress 2

1) As the coefficient of Ambient reflection (K_a) lies between 0 and 1, the lesser value (value closer to 0) conveys the message that the surface is more absorbing in nature and similarly the surfaces having values of K_a closer to 1, are highly reflective.

2) As Black hole in universe reflects nothing so the value of K_a should be 0.

Check Your Progress 3

1) As the surface under exposure transforms from imperfect reflector to the perfect one, the respective viewing angle of reflection keeps on decreasing because the light reflected will be having more precise direction.

2) Since no variation in intensity of light is observed in any direction, it implies the surface is rough and it which leads to diffused reflection. If the surface is smooth and shiny then reflection should be in one particular direction and not uniform in all directions.

3) Lambert's Law forms the basis of the Lambertion reflection; the law says that intensity of light is directly proportional to the angle between incident ray and the normal.

Check Your Progress 4

1) Lesser the value of Specular reflection parameter conveys the message that the surface under exposure is quite dull, the value of n increases as the surface shines more and more.

2) Gourand Shading.

3) Refer to Shading (8.4).

4) Say < refers to increasing order of comparative performance then constant shading < Gourand Shading < Phong Shading for reason behind refer to merits and demerits of each type of shading.

5) Refer to shading section.

6) Phong shading is better than Gourand shading, the reason being that in Gouraud Shading ∵ of the consideration of average normal, the intensity is uniform across the edge between two vertices and this causes problem in shading. Deficiencies of Gourand Shading are overcome in Phong Shading, where we are finding intensities at different points across the edge and it is observed that intensity is varying non- uniformly across the edge between two vertex points, but this variation is uniform in Gouraud Shading. Thus Phong shading gives much better effect.

7) Resulting intensity (I) = $I_a K_a + I_d K_d \cos\theta + I_s K_s \cos^n \alpha$;

8) When an object is under the exposure of light, then the rays of light are distributed over the surface and the distribution of intensity pattern depends very much on the shape of object. Now to represent such 3D scene on computer we need to do rendering, i.e., transforming the 3D image into a 2D image, and because of this rendering there is a possibility of loss of information like depth, height etc., of an object in the scene. So to preserve this takes different Illumination models into consideration, for preserving the information embedded in 3D scene and let it not be lost while transforming it into 2D scene.

Check Your Progress 5

1) Ray tracing is a method of generating realistic images by computer, in which the paths of individual rays of light are followed from the viewer to their points of origin". Whereas ray casting only concerns itself with finding the visible surfaces of objects, ray tracing takes that a few steps further and actually tries to determine what each visible surface looks like.

2) Ray tracing makes use of the actual physics and mathematics behind light. Thus, the images produced can be strikingly life-like, or "photo-realistic". For more details refer to ray tracing algorithm.

3) If you put a grid over the image and only color those squares that are entirely within the circle, you will get something like next figure. This blockiness is called "aliasing", and is exactly what happens when you try to display a circle on a computer screen. The problem may be simplified by using a finer grid. Anti-aliasing is a method for improving the realism of an image by removing the jagged edges from it.

UNIT 1 COMPUTER ANIMATION

Structure **Page Nos.**

1.0 Introduction 5
1.1 Objectives 5
1.2 Basics of Animation 6
  1.2.1 Definition 7
  1.2.2 Traditional Animation Techniques 7
  1.2.3 Sequencing of Animation Design 9
  1.2.4 Types of Animation Systems 11
1.3 Types of Animation 14
1.4 Simulating Accelerations 15
1.5 Computer Animation Tools 20
  1.5.1 Hardware 20
  1.5.2 Software 21
1.6 Applications for Computer Animation 23
1.7 Summary 27
1.8 Solutions/Answers 28

1.0 INTRODUCTION

The word Animation is derived from 'animate' which literally means 'to give life to', 'Animating' a thing means to impart movement to something which can't move on its own.

In order to animate something, the animator should be able to specify, either directly or indirectly, how the 'thing' is to move through time and space. We have already discussed various transformations in Block 2 Unit 1 using which you can impart motion, size alteration, rotation, etc, to a given graphic object. Before dealing with complexities of animation, let us have a look at some basic concepts of Animation in section 1.2. In section 1.3, we will discuss different kinds of animations.

In our childhood, we all have seen the flip book of cricketers which came free along with some soft drink, where several pictures of the same person in different batting or bowling actions are intact sequentially on separate pages, such that when we flip the pages of the book the picture appears to be in motion, this was a flipbook (several papers of the same size with an individual drawing on each paper so the viewer could flip through them). It is a simple application of the basic principle of Physics called Persistence of Vision. This low tech animation was quite popular in the 1800s when the Persistence of vision (which is 1/16 th of a second) was discovered. This discovery led to some more interesting low tech animation devices like, the zoetrope, wheel of life, etc. Later, depending on many basic mathematics and physics principles, several researches were conducted which allowed us to generate 2D/3D animations.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- describe the basic properties of animation;
- classify the animation and its types;
- discuss how to impart acceleration in animation, and
- give examples of different animation tools and applications.

1.2 BASICS OF ANIMATION

Traditional and historical methods for production of animation

In units 1 and 2 of block 2 we have studied the transformations involved in computer graphics but you might not have noticed there that all transformations are related to space and not to time. Here, lies the basic difference between Animation and graphics. The difference is that animation adds to graphics, the dimension of time, which vastly increases the amount of information to be transmitted, so some methods are used to handle this vast information and these methods are known as animation methods the *Figure 1* gives a broad description of methods of animation.

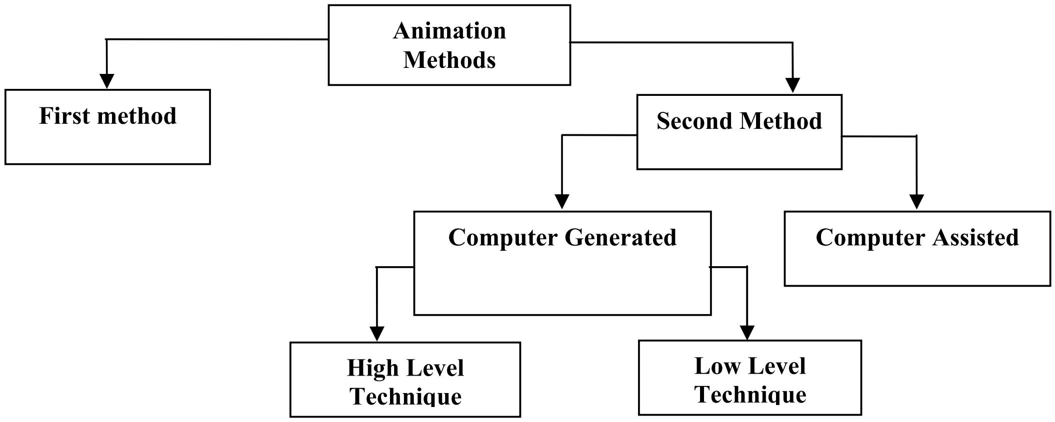


Figure 1: Methods of animation

First method: Here, artist creates a succession of cartoon frames, which are then combined into a film.

Second method: Here, the physical models are positioned to the image to be recorded. On completion the model moves to the next image for recording and this process is continued. Thus, the historical approach of animation has classified computer animation into two main categories:

a) *Computer-assisted animation* usually refers to 2D systems that computerise the traditional animation process. Here, the technique used is interpolation between key shapes which is the only algorithmic use of the computer in the production of this type of animation equation, curve morphing (key frames, interpolation, velocity control), image morphing.

b) C*omputer generated animation* is the animation presented via film or video, which is again based on the concept of persistence of vision because the eye-brain assembles a sequence of images and interprets them as a continuous movement and if the rate of change of pictures is quite fast then it induce the sensation of continuous motion.

This motion specification for computer-generated animation is further divided into 2 categories:

Low level techniques (motion specific)

Techniques used to fully control the motion of any graphic object in any animation scene, such techniques are also referred as motion specific techniques because we can specify the motion of any graphic object in scene, techniques like interpolation, approximation etc., are used in motion specification of any graphic object. *Low level*

techniques are used when animator usually has a fairly specific idea of the exact motion that he or she wants.

High level techniques (motion generalized)

Techniques used to describe general motion behavior of any graphic object, *these techniques are* algorithms or models used to generate a motion using a set of rules or constraints. The animator sets up the rules of the model, or chooses an appropriate algorithm, and selects initial values or boundary values. The system is then set into motion and the motion of the objects is controlled by the algorithm or model, this approaches often rely on fairly sophisticated computation such as vector algebra and numerical techniques and others.

Isn't it surprising that *the Computer animation has been around as long as computer graphics* which is used to create realistic elements which are intermixed with the live action to produce animation. The traditional way of animation is building basis of the computer generated animation systems and are widely used now a days by different companies like, Disney, MGM, Warner Bros, etc, to produce realistic 3D animation using various animation tools. As various tools are available for different uses. Thus, the basic problem is to select or design animation tools which are expressive enough for the animator to specify what s/he wants to specify while at the same time are powerful or automatic enough that the animator doesn't have to specify the details that s/he is not interested in. Obviously, there is no single tool that is going to be right for every animator, for every animation, or even for every scene in a single animation. The appropriateness of a particular animation tool depends on the effect desired by the animator. An artistic piece of animation will probably require different tools for an animation intended to simulate reality. Some examples of the latest animation tools available in the market are Softimage (Microsoft), Alias/Wavefront (SGI), 3D studia MAX (Autodesk), Lightwave 3D (Newtek), Prism 3D Animation Software (Side Effects Software), HOUDINI (Side Effects Software), Apple's Toolkit for game developers, Digimation, etc.

After having some breifings about the overall topic of animation, now let us go to its details. Firstly we define/describe computer animation which is a time-based phenomenon that covers any change of appearance or any visual effect with respect to the time domain, which includes motion, i.e., positional change(translation/rotation), time-varying changes in shape, colour (palette), transparency and even changes of the rendering technique.

1.2.1 Definition

A time based phenomenon for imparting visual changes in any scene according to any time sequence, the visual changes could be incorporated through translation of object, scaling of object, or change in colour, transparency, surface texture etc.

Note: It is to be noted that computer animation can also be generated by changing camera parameters such as its position, orientation, focal length etc. plus changes in the light effects and other parameters associated with illumination and rendering can produce computer animation too.

1.2.2 Traditional Animation Techniques

Before the advent of computer animation, all animation was done by hand, which involves an enormous amount of work. You can have an idea of work by considering that each second of animation film contains 24 frames (film) then, one can imagine the amount of work in creating even the shortest of animated films. Before, creating any animation the first step is to design the concerned storyboard which is the first sight of what a cartoon or a piece of animation is going to look like. It appears as a

series of strip comics, with individual drawings of story lines, scenes, characters, their emotions and other major part of movie. Now, let us discuss a couple of different techniques, which were developed for creating animation by hand.

Key Frames

After a storyboard has been laid out, the senior artists go and draw the major frames of the animation. These major frames are frames in which a lot of change takes place. They are the key points of the animation. Later, a bunch of junior artists draw in the frames in between. This way, the workload is distributed and controlled by the key frames. By doing work this way, the time in which an animation can be produced is cut dramatically, depending on the number of people working on the project. Work can be done simultaneously by many people, thus cutting down the time needed to get a final product out.

Cel Animation

By creating an animation using this method, each character is drawn on a separate piece of transparent paper. A background is also drawn on a separate piece of opaque paper. Then, when it comes to shooting the animation, the different characters are overlaid on top of the background in each frame. This method also saves time in that the artists do not have to draw in entire frames, but rather just the parts that need to change such as individual characters, even separate parts of a character's body are placed on separate pieces of transparent paper. For further understanding, let us have an example. Say you want to show that an aeroplane is flying. You can draw an aeroplane on a transparent sheet and on another opaque sheet you can have clouds. Now, with the opaque sheet as background you can move the transparent sheet over it, which gives the feeling of flying aeroplane.

These traditional techniques were extended to the era of computer animation techniques and hence different animation systems are evolved. We cannot say which technique is better because different techniques are used in different situations. In fact all these animation techniques are great, but they are most useful when all of them are used together. Cel animation by itself would not help out much if it wasn't for key frames and being able to distribute the workload across many people.

Now, let us also discuss the computer animation methods, which are in wide use, two of the typical computer animation methods are 'frame' animation and sprite animation.

Frame animation non- interactive animation rectangular shape (Cartoon movies)

This is an "internal" animation method, i.e., it is animation inside a rectangular frame. It is similar to cartoon movies: a sequence of frames that follow each other at a fast rate, fast enough to convey fluent motion. It is typically pre-compiled and non-interactive. The frame is typically rectangular and non-transparent. Frame animation with transparency information is also referred to as "**cel"** animation. In traditional animation, a cel is a sheet of transparent acetate on which a single object (or character) is drawn.

Sprite animation interactive, may be non rectangular (Computer games)

In its simplest form it is a 2D graphic object that moves across the display. Sprites often can have transparent areas. Sprites are not restricted to rectangular shapes. Sprite animation lends itself well to be interactive. The position of each sprite is controlled by the user or by an application program (or by both). It is called "external" animation.We refer to animated objects (sprites or movies) as "animobs". In games

and in many multimedia applications, the animations should adapt themselves to the environment, the program status or the user activity. That is, animation should be *interactive*. To make the animations more event driven, one can embed a script, a small executable program, in every animob. Every time an animob touches another animob or when an animob gets clicked, the script is activated. The script then decides how to react to the event (if at all). The script file itself is written by the animator or by a programmer.

☞ Check Your Progress 1

1) What do you mean by animation, what are different ways to produce it?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

2) What do you mean by computer generated and computer assisted animations?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

3) Differentiate between
   a) Key frame and Cel animation b) Low level and high-level animation techniques.

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

4) Which animation technique is better, Keyframe or Cel animation?

…………………………………………………………………………………

…………………………………………………………………………………

…………………………………………………………………………………

1.2.3 Sequencing of Animation Design

Till now we have discussed a lot about the traditional and current trends of computer generated animation but now it, is time to practically discuss the necessary sequencing of animation steps which works behind the scenes of any animation.

This sequencing is a standard approach for animated cartoons and can be applied to other animation applications as well. General Steps of designing the animation sequence are as follows:

1) **Layout of Storyboard:** Storyboard layout is the action outline used to define the motion sequence as a set of basic events that are to take place. It is the type of animation to be produced which decides the storyboard layout. Thus, the storyboard consists of a set of rough sketches or a list of basic ideas for the motion.

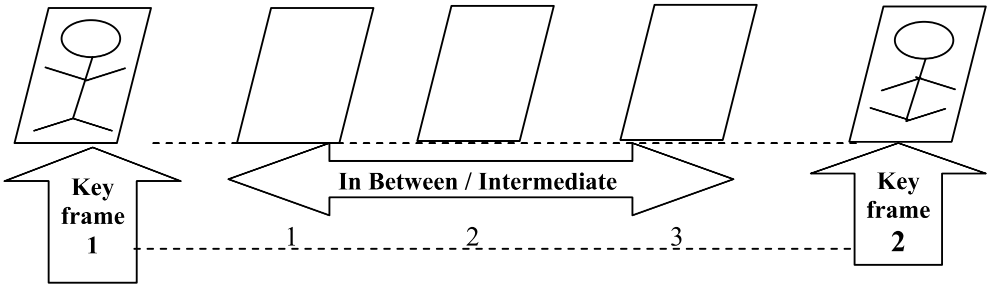
2) **Definition of Object:** The object definition is given for each participant object in action. The objects can be defined in terms of basic shapes, associated movements or movement along with shapes.

3) **Specification of Key Frame:** It is the detailed drawing of the scene at a certain time in the animation sequence. Within each key frame, each object is positioned according to time for that frame. Some key frames are chosen at the extreme positions in the action; others are spaced so that the time interval between key frames is not too great. More key frames are specified for intricate motion than for simple, slowly varying motions.

4) **In-between frames Generation:** In-between frames are the intermediate frames between the key frames. The number of in between frames is dependent on the media to be used to display the animation. In general, film requires 24 frames per second, and graphic terminals are refreshed at the rate of 30 to 60 frames per second. Typically the time interval for the motion are set up so that there are 3 to 5 in-betweens for each pair of key frames. Depending upon the speed specified for the motion, some key frames can be duplicated.

Note: There are many applications that do not follow this sequence like, real time computer animations produced by vehicle driving or flight simulators, for instance, display motion sequence in response to setting on vehicle or aircraft controls, plus the visualization applications are generated by the solution of numerical models. And for frame-by-frame animation each frame of the scene is separately generated and stored. Later the frames can be recorded on film or they can be consecutively displayed in "real time playback" mode.

In order to explain the overall process of animation mathematically consider the *Figure 2*. In order to have smooth continuity in motion of objects, a number of in-between frames are sandwiched between two key frames Now, there is a relation between different parameters, i.e., key frame, in-between frame, time, number of frames required per second which is



Formula:
Required Key frames for a film= $\dfrac{\{[\text{Time(in seconds)}]*[\text{frames required per second(in general =24)}]\}}{\{\text{no. of in between frames}\}}$

Figure 2

Example 1: How many key frames does a one-minute animation film sequence with no duplications require ?

Solution: One minute = 60 seconds
No. of frames required per second=24
No. of frames required in entire film=24*60=1440

That is we would need 1440 frames for a one-minute animation film.

Example 2: How many key frames does a one-minute animation film sequence with no duplications require if there are five in betweens for each pair of key frames ?

Solution: One minute = 60 seconds

No. of frames required per second = 24

No. of in-between frames = 5

No. of frames required in entire film=(24*60)/5=288

That is we would need 288 key frames for a one-minute animation film if the number of in-between frames is 5.

☞ Check Your Progress 2

1) How many frames does a 30-second animation film sequence with no duplication require? What will be the answer if duplication is there?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) How many frames does the same film as in E1 require if it has three in-between frames?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) Why does an animation film require 24 animation frames per second? Can this number be less than 24? If yes, then up to what extent this number can decrease and what will be the effect of animation on the reduction of this number?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

4) What are the steps to create an animation?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

1.2.4 Types of Animation Systems

We have discussed above that the sequencing of animation is useful in developing any animation. This sequencing is more or less the 'same in all animation systems'. Before proceeding to the types of animation in the next section, let us study the types of Animation Systems.

So let us discuss a few animation systems, which are generally used:

Key Frame Systems

This technique is for low-level motion control. Actually these systems include languages which are designed simply to generate the in-betweens from the user-specified key frames.

Usually, each object in the scene is defined as a set of rigid bodies connected at the joints and with a limited number of degrees of freedom. Key frame systems were developed by classical animators such as Walt Disney. An expert animator would design (choreograph) an animation by drawing certain intermediate frames, called Key frames. Then other animators would draw the in-between frames.

The sequence of steps to produce a full animation would be as follows:

1) Develop a script or story for the animation.
2) Lay out a storyboard, that is a sequence of informal drawings that shows the form, structure, and story of the animation.
3) Record a soundtrack.
4) Produce a detailed layout of the action.
5) Correlate the layout with the soundtrack.
6) Create the "key frames" of the animation. The key frames are those where the entities to be animated are in positions such that intermediate positions can be easily inferred.
7) Fill in the intermediate frames (called "in-betweening" or "tweening").
8) Make a trial "film" called a "pencil test".
9) Transfer the pencil test frames to sheets of acetate film, called "cels". These may have multiple planes, e.g., a static background with an animated foreground.
10) The cels are then assembled into a sequence and filmed.

With computers, the animator would specify the key frames and the computer would draw the in-between frames ("tweening"). Many different parameters can be interpolated but care must be taken in such interpolations if the motion is to look "real". For example, in the rotation of a line, the angle should be interpolated rather than the 2D position of the line endpoint. The simplest type of interpolation is linear, i.e., the computer interpolates points along a straight line. A better method is to use cubic splines for interpolation (which we have studied in Block 3). Here, the animator can interactively construct the spline and then view the animation.

Note: From the above discussion, it is clear that in key frame systems the in-between frames can be generated from the specification of two or more key frames, and among them we can set the motion path of the object under consideration by describing its kinematics description as a set of spline curves. For complex scenes we can separate the frames into individual components or objects called cels (Celluloid transparencies). In these complex scenes, we can interpolate the position of individual objects between any two times. And in this interval the complex objects in the scene may suffer from various transformations like the shape or size of object may change over time, etc., or the entire object may change to some other object. These transformations in a key frame system lead to Morphing, Zooming, Partial motion, **Panning** (i.e., shifting of background/foreground to give the illusion that the camera seems to follow the moving object, so that the background/ foreground seems to be in motion), etc.

MORPHING: Transformation of object shapes from one form to another is called **morphing** (short form of metamorphism). Morphing methods can be applied to any motion or transition involving a change in shape. To understand this, consider the *Figure 3*.

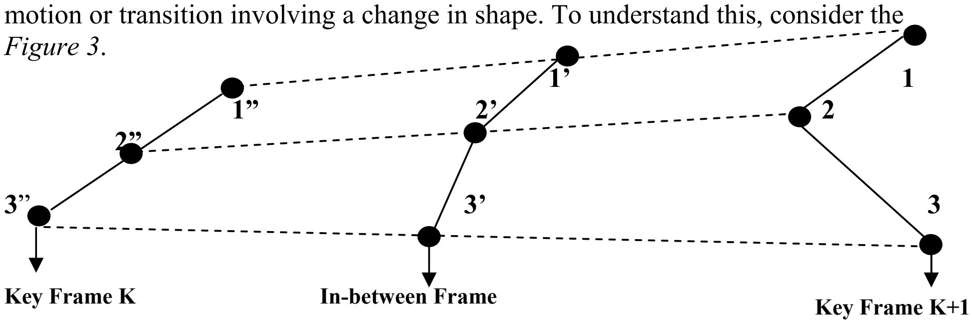


Figure 3

The *Figure 3* shows the linear interpolation for transformation of two connected line segments in key frame K on to a line segment in key frame K+1. Here, the number of vertices in both frames are the same, so simply position changes.

But if the situation is *vice versa*, i.e., say key frame K is a Line and key frame K+1 is a pair of lines, we need to have linear interpolation for transforming a line segment in key frame K into two connected line segments in key frame K+1. As transformation is from one line to two lines and since frame K+1 has extra vertex we add a vertex between 1 and 3 in frame K, to have balance between number of vertices and edges in two frames

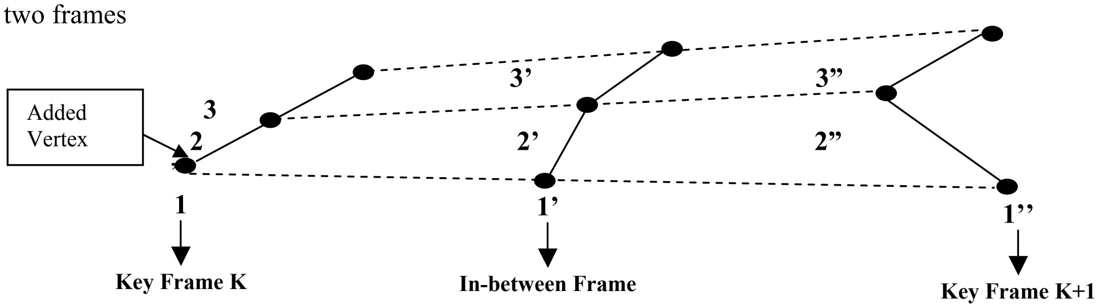


Figure 4

Scripting Systems are the earliest type of motion control systems. Scripting systems allow object specifications and animation sequence to be defined with a user input script., and from this script, a variety of various objects and motions can be constructed. So, to write the script the animator uses any of the scripting languages. Thus, the user must learn this language and the system. Some scripting systems are PAWN (An embedded scripting language formerly called Small) with syntax similar to C, ASAS (Actor Script Animation Language), which has a syntax similar to LISP. ASAS introduced the concept of an actor, i.e., a complex object which has its own animation rules. For example, in animating a bicycle, the wheels will rotate in their own coordinate system and the animator doesn't have to worry about this detail. Actors can communicate with other actors by sending messages and so can synchronize their movements. This is similar to the behavior of objects in object-oriented languages.

Parameterised Systems these are the systems that allow objects motion characteristics to be specified as part of the object definitions. The adjustable parameters control such objects characteristics as degree of freedom, motion limitations, and allowable shape changes.

☞ Check Your Progress 3

1) After being exposed to different concepts of computer graphics and animation through and Block 2 (which inform uses concepts of CS-60) and the introduction of this Unit, can you answer one simple question. "*when do we need to use computer graphics in computer animation ?*"

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2) What do you think which type of animation system will be suitable to generate cartoon films and which one will be suitable to generate computer games?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

3) What are animobs, in which system of animation they are used?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

4) What do we mean by Morphing and Panning? What is their significance in animation?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

1.3 TYPES OF ANIMATIONS

Procedural Animation: This type of animation is used to generate real time animation, which allows a more diverse series of actions to happen. These actions be created using some could otherwise predefined animation procedures are used to define movement over time. There might be procedures that use the laws of physics (Physical i.e., modeling based) or animator-generated methods. Some example of procedural animation is collision which is an activity that is the result of some other action (this is called a "secondary action"), for example throwing a ball which hits another object and causes the second object to move; simulating particle systems (smokes water etc.) hair and for dynamics. In computer video games it is often used for simple things like players head rotation to look around, etc.

Representational Animation: This technique allows an object to change its shape during the animation. There are three sub-categories to this. The first is the animation of articulated objects, i.e., complex objects composed of connected rigid segments. The second is soft object animation used for deforming and animating the deformation of objects, e.g., skin over a body or facial muscles. The third is morphing which is the changing of one shape into another quite different shape. This can be done in two or three dimensions.

Stochastic animation: This uses stochastic processes (A stochastic process can be considered as a random function). This randomness could be in time or space variable of function, the randomness in time leads to stochastic animation to control groups of objects, such as in particle systems. Examples are fireworks, fire, waterfalls, etc., or speech audio signal, medical data ECG, BP, etc, or Random walk.

Behavioural animation: Used to control the motion of many objects automatically. Objects or "actors" are given rules about how they react to their environment. The primary difference is in the objects being animated, instead of simply procedurally controlling the position of tiny objects. This type of animation is generally used to animate flocks, school, herds and crowds. Examples are schools of fish or flocks of

birds where each individual behaves according to a set of rules defined by the animator.

So as to generate these types of animations, we need to have familiarisation with some general functions which every animation software is suppose to have. In general animation functions include a graphic editor, a key frame generator, an in-between generator, and standard graphic routines. The graphic editor allows us to design and modify object shapes using splines surfaces, Constructive Solid Geometry (CSG) methods and other representational schemes.

In the development of an animation sequence some steps are well suited for computer solutions, these include object manipulations, rendering, camera motions and the generation of in-betweens. Animation packages such as wave front provide special functions for designing the animation and processing individual objects.

Some general functions available in animation packages are:

- Object Function to store and manage the object database, where the object shapes and associated parameters are stored and updated in the database.

- Object Function for motion generation and object rendering. Motions can be generated according to specified constraints using 2D and 3D transformations. Standard functions can then be applied to identify visible surfaces and apply the rendering algorithms.

- Object function to simulate camera movements, standard motions like, zooming, panning, tilting etc. Finally the specification for the key frames, the in-between frames can be automatically generated.

1.4 SIMULATING ACCELERATIONS

In Block 2, we have seen the dominance and role of mathematics in computer graphics and here, we will undertake the involvement of mathematics to simulate motion. As the motion may be uniform with acceleration to be zero, positive or negative or non-uniform, the combination of such motions in an animation contributes to realism. To impart motion to a graphic object, curve fittings are often used for specifying the animation paths between key frames. Given the vertex positions at the key frame, we can fit the positions with linear or non-linear paths, which determines the trajectories for the in-between and to simulate accelerations, we can adjust the time spacing for the in-betweens.

Let us discuss different ways of simulating motion:
- Zero Acceleration (Constant Speed)
- Non-Zero Accelerations
  - Positive accelerations
  - Negative accelerations or Decelerations
  - Combination of accelerations

1) **Zero Acceleration (Constant Speed)**: Here, the time spacing for the in-betweens (i.e., in-between frames) is at equal interval; i.e., if, we want N in-betweens for key frames at time Ta and Tb, then, the time interval between key frames is divided into N+1 sub-intervals leading to in-between spacing of Δ T given by the expression in *Figure 5*.

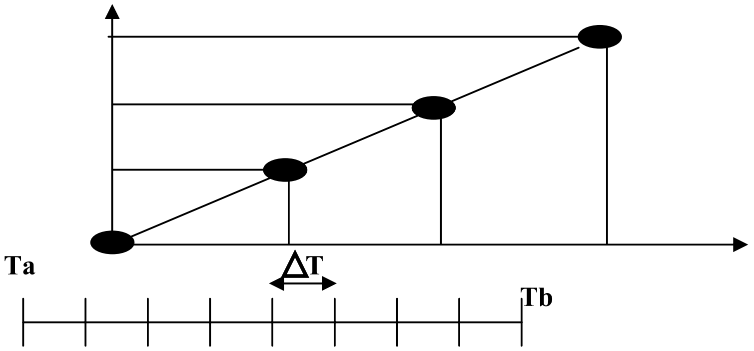


Figure 5

Thus, the time of any J th in-between can be found by

$$Tj = Ta + J^* \Delta T \qquad J = 1, 2, 3, \ldots\ldots., N$$

Note: A linear curve leads to zero acceleration animation.

2) **Non-Zero Accelerations**: This technique of simulating the motion is quite useful introducing the realistic displays of speed changes, specially at the starting and completion of motion sequence. To model the start-up and slow-down portions of an animation path, we use splines or trigonometric functions (note: trigonometric functions are more commonly used in animation packages, whereas parabolic and cubic functions are used in acceleration modeling).

   - **Positive Accelerations:** In order to incorporate increasing speed in an animation the time spacing between the frames should increase, so that greater change in the position occur, as the object moves faster. In general, the trigonometric function used to have increased interval size the function is (1-Cos Θ) ,0<Θ<Π/2 .

     For n in-betweens, the time for the J th in-between would be calculated as

$$\Delta T_J = Ta + \Delta T[1 - Cos(J\Pi/2(N+1))] \qquad J = 1,2,3,\ldots\ldots..,N$$

△T=time difference between two key frames =Tb-Ta

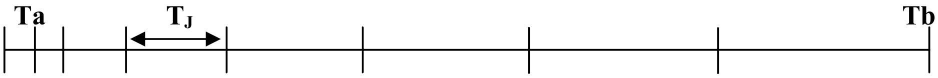


Figure 6

Figure 6 represents a positive acceleration case because the space (i.e., time space) between frames continuously increases leading to the increase in accelerations i.e., changes in object position in two frames is fast. Let us, study the trigonometric function used to achieve positive acceleration, i.e., Y=(1-Cos Θ) ,0<Θ<Π/2

$$\text{At}\,\Theta = 0; \quad Y = (1 - \text{Cos}0) = 1 - 1 = 0$$
$$\text{At}\ \Theta = \Pi/2; \quad Y = \ (1 - \text{Cos}\,\Pi/2) = 1 - 0 = 1$$

Now look at *Figure 7* for proper understanding of concept.

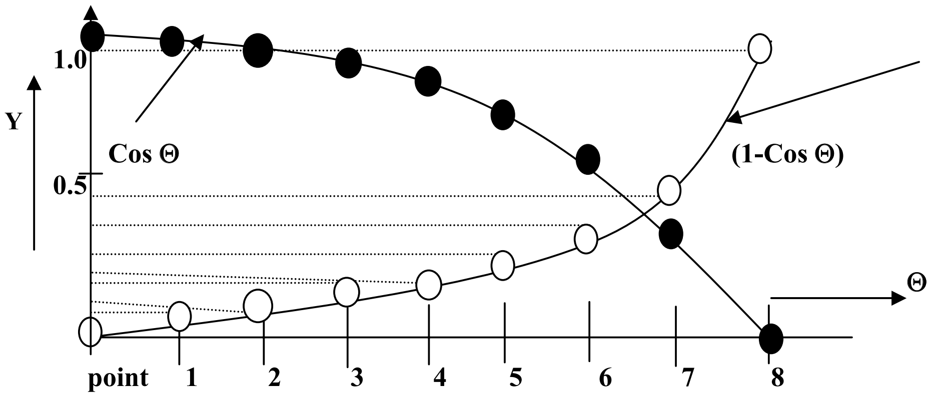


Figure 7

Note: Having projections of points on curve, over Y axis, we will receive a pattern similar to *Figure 6*, which is required to produce positive acceleration.

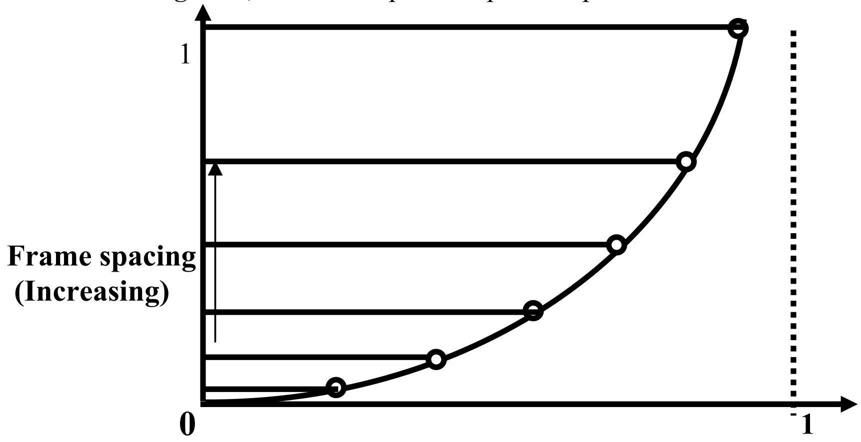


Figure 7(a)

Increases in gap along y-axis depict that spacing between key frames increases, which leads to accelerated motion.

As our aim here is to have acceleration in the motion so we create N-in between frames, between two key frames (which leads to N+1 sections) and divide Θ axis in to N fragments, for each fragment, we find Y=(1-CosΘ). Substituting different values of Θ we get different Y as shown in *Figure 7 and 7(a)*, the space between frames is continuously increasing, imparting an accelerated motion.

Length of each subinterval ($\triangle\Theta$) = (Θ_1-Θ_2)/no. of subintervals

$$= (\Pi/2 - 0)/N + 1 = \Pi/2(N + 1)$$

Hence, change in ($\triangle\Theta$) produces change of 1- Cos (Π / 2(N+1))

- **Negative Accelerations:** In order to incorporate decreasing speed in an animation the time spacing between the frames should decrease, so that there exist lesser change in the position as the object moves. In general, the trigonometric function used to have increased interval size the function is Sin Θ ,0<Θ<Π/2.

  For n in-betweens, the time for the J th in between would be calculated as:

$$T_J = Ta + \Delta T[Sin(J\,\Pi/2(N+1))] \qquad J = 1,2,3,......N$$

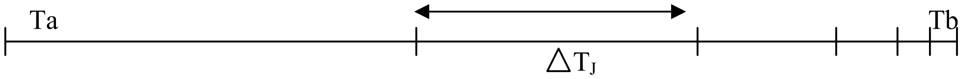


Figure 8

As in the *Figure*, the spacing between frames is decreasing so the situation changes from fast motion to slow motion, i.e., decreasing acceleration or deceleration. Let us, study the trigonometric function used to achieve this negative acceleration, i.e.,

Y=SinΘ in the interval $0<\Theta<\Pi/2$

$$\text{At}\,\Theta = 0 \quad ; \quad Y = \text{Sin}(\Theta = 0) = 0$$
$$\text{At}\,\Theta = \Pi/2 \quad ; \quad Y = \text{Sin}(\Theta = \Pi/2) = 1$$

Now, dividing the Θ range in to N+1 parts and plotting the graph Y Vs Θ we will get a sine curve as shown below:

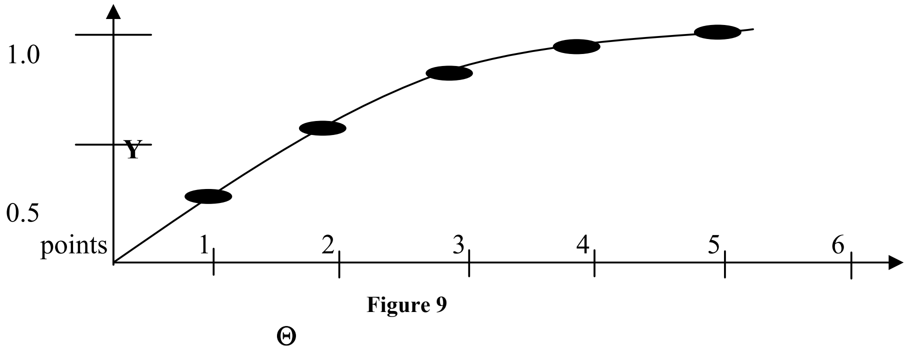


Figure 9

Note: Having projections of points on curve, over Y axis we will receive a pattern similar to *Figure 8*, which is required to produce negative acceleration.

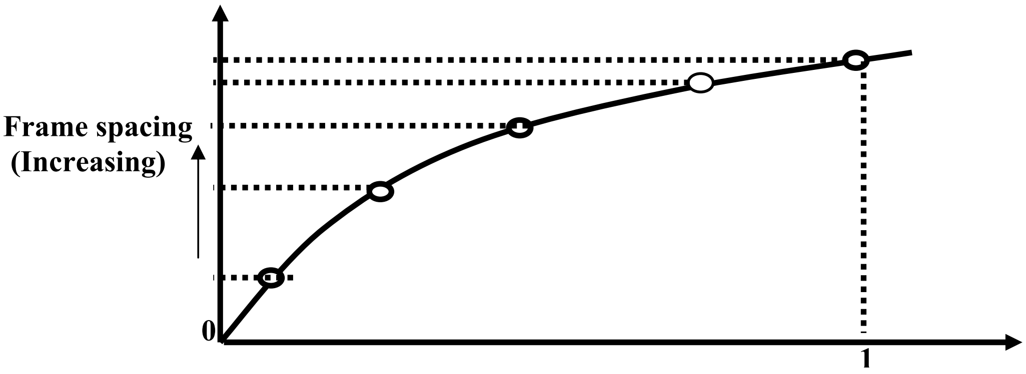


Figure 9(a)

Having projections on y-axis we find, as we, move from y = 0 to y = 1 the distance or spacing between frames continuously decreases leading to negative acceleration or slow motion.

The range $0<\Theta<\Pi/2$ is divided into N+1 parts so length of each fragment will be
$\Delta\Theta) = (\Pi/2 - 0)/N + 1 = \Pi/2(N + 1)$

Each $\Delta\Theta$ change produces change of $\Delta Y = \text{Sin}(\Pi/2(N+1))$
For Jth segment $\Delta Y = \text{Sin}(J * \Pi/2(N+1))$

Therefore, out of N in between frames the time for the Jth in-between frame is
$T_J = Ta + \Delta T[\text{Sin}(J\,\Pi/2(N+1))]$

Where ΔT=Gap between two key frames=Tb-Ta

- **Combination of Positive and Negative accelerations:** In reality, it is not that a body once accelerated or decelerated will remain so, but the motion may contain both speed-ups and slow-downs. We can model a combination of accelerating – decelerating motion by first increasing the in-between spacing and then decreasing the same. The trigonometric function used to accomplish these time changes is

  $Y = (1 - \text{Cos}\Theta)/2 \quad ; \quad 0 < \Theta < \Pi/2$

  The time for the Jth in-between is calculated as

$$T_J = Ta + \Delta T\{[1-(Cos(J\,\Pi/(N+1))]/2\} \qquad J = 1,2,3,........N$$

Δ T=time difference between two key frames =Tb-Ta

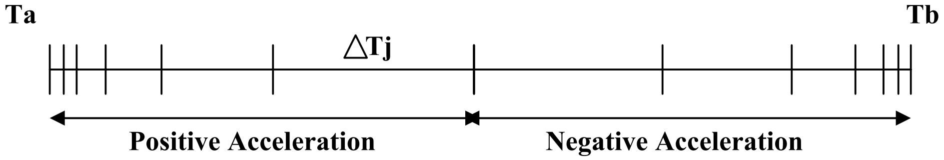


Figure 9

In the shown *Figure 9*, the time interval for the moving object first increases, then the time interval decreases, leading to the simulation of motion that first accelerates and then decelerates.

Note: Processing the in-betweens is simplified by initially modeling "skeleton" (wire frame) objects. This allows interactive adjustment of motion sequences. After the animation sequence is completely defined, the objects can be fully rendered.

☞ Check Your Progress 4

1) What type of acceleration will be simulated if:
   a) Distance between frames is constant
   b) Distance between frames continuously increases
   c) Distance between frames continuously decreases

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

2) What type of animation does a straight line function (y=mx+c) produce and why?

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

   …………………………………………………………………………………

3) Why the animation seems to be accelerating if the spacing between frames keeps on increasing?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

Also discuss the case when it is decelerating the spacing between frames keep on decreasing.

1.5 COMPUTER ANIMATION TOOLS

To create different types of animation discussed above, we, need to have special software and hardware too. Now, the basic constraint is about the choice of proper hardware and software out of the many available in the market.

Thus, the basic problem is to select or design animation tools, which are expressive enough for the animator to specify what s/he wants to specify which, at the same time, are powerful or automatic enough so that the animator doesn't have to specify the details that s/he is not interested in. Obviously, there is no single tool that is going to be right for every animator, or for every animation or even for every scene in a single animation. The appropriateness of a particular animation tool depends on the effect desired by the animator. An artistic piece of animation will probably require different tools (both software and hardware) to simulate reality. Along with software we need to have some special hardware to work with the concerned software.

Here is a short list of some 3D animation software:

Softimage (Microsoft)
Alias/Wavefront (SGI)
3D studia MAX (Autodesk)
Lightwave 3D (Newtek)
Prism 3D Animation Software (Side Effects Software)
HOUDINI (Side Effects Software)
Apple's Toolkit for game developers
Digimation etc.

The categories of both hardware and software required to work on animation are now to be discussed. Computer animation can be done on a variety of computers. Simple cell animation requires nothing more than a computer system capable of simple graphics with proper animation software. Unfortunately, most of the computer animation that you see on television and in other areas is done on extremely sophisticated workstations. So as to cover both hardware and software required for animation, the tools are seggregated into two sections, software and hardware.

In the hardware section, all the different computer platforms on which computer animation is done are explained. The software is explained in the software section. Only the most popular and most well known software are explained, since, a huge number of software are available in the market, so it would be practically impossible to name all computer animation programs because there are so many of them.

1.5.1 Hardware

Hardware comes in many shapes, sizes, and capabilities. Some hardware is specialised to do only certain tasks. Other kinds of hardware do a variety of things. The following are the most common hardware used in the field of computer animation.

SGI (Silicon Graphics Inc.): The SGI platform is one of the most widely used hardware platforms in professional or broadcast quality computer animation productions. Some of the salient features of SGI computers are that, of the different types available in the market. SGI computers are extremely fast, produce excellent results, and operate using the widespread UNIX operating system. SGIs are produced by Silicon Graphics, ranging from the general purpose Indy®, to the high power Indigo2 Extreme® used to produce animations, the Onyx®, which is especially suited to do complex calculations. Almost all major production studios use SGIs state of the art software like Wavefront, Alias, and SoftImage which run on SGIs.

PCs (Personal Computers really): PCs are really versatile machines, which have been around for years. PCs are the favourite of many computer users, because of their combination of flexibility and power, PC's have proven to be very useful for small companies and other businesses as platforms to do computer animation. Some applications such as 3DStudio and Animator Studio are used on PCs to make animations. PCs are relatively cheap and provide pretty good quality of animation. Recently though, PCs have been getting a lot of attention from different production houses because of their relatively small price and the quality of the finished products.

Amiga: Originally owned by Commodore, Amiga computers have held a position in the computer animation industry for a number of years. It is widely used in introduced effects and amination for movies/TV shows etc. There are two software packages that Amiga's are basically known for: Video Toaster and LightWave 3D. The Amiga is based on Commodore, but it has been greatly customised to be a graphics machine.

Macintosh: Macs were originally designed to be graphic and desktop publishing machines. Macs did not become that widely known until recently, when newer, faster models came out. Many people consider Macs slow and inefficient, but that is not necessarily true. Right now with the advent of the Power Macintosh, the Mac is a pretty useful tool for small-scale companies wishing to do nice looking applications. Many companies are producing computer graphics and animation software for Macintosh. Some of these are Adobe with products such as Photoshop and Premiere and Strata with Strata Studio Pro. There are also a few applications that were ported to the Macintosh from the SGIs such as Elastic Reality and Alias Sketch (a lower end version of Alias). Lately, a lot of production studios have started using Macs because of their graphical abilities for smaller scale projects.

1.5.2 Software

You might have the best hardware in the world, but without a good software package, your hardware can do nothing. There are literally hundreds of computer animations and graphics software packages out there, however, only a few are considered industry favourites. Some of the most popular software packages used by companies, schools, and individuals all around the globe are:

1) **Adobe flash:** Formerly, it was known as Macromedia flash and prior to this, it was Futuresplash. It is in fact an IDE that refers to both Adobe flash player and the multimedia authoring program which are used to create applications like, websites, games, movies, etc. It has features to support both vector and raster graphics, the scripting language used with it is k.a Action script. So, Adobe flash is an IDE (integrated development environment). Flash players is important component of Adobe flash because, it is the virtual machine which is used to run or parse the flash files (traditionally flash files are called flash movies that have software extension). Now-a-days the flash technology is used very frequently to create interactive websites, Animation, Advertisements, etc.

2) **3Dstudio:** 3DStudio is a 3D computer graphics programme. 3DStudio runs on PCs. It is relatively easy to use. Many schools and small time production studios

use 3DStudio to satisfy their needs. 3DStudio is created by AutoDesk. 3DStudio consists of a 2D modeler, in which shapes can be drawn, a 3D Lofter, in which 2D shapes can be extruded, twisted, or solidified to created 3D objects. Then, there is a 3D modelet, in which a scene is created. Finally, there is an animator in which key frames are assigned to create an animation and a material editor, in which a great variety of textures can be created. Overall, this is a great program.

3) **3DStudio Max:** The successor to 3DStudio 3.0, 3DStudio Max runs under WindowsNT. It is entirely object oriented, featuring new improvements such as, volumetric lighting, space warps, and an all new redesigned interface.

LightWave3D

LightWave 3D is another high end PC 3D computer graphics software package. Originally developed for the Amiga platform, LightWave 3D is now also available on the PC. LightWave 3D is used in quite a few television productions such as, Babylon 5 and SeaQuest.

Adobe Photoshop

Although Adobe Photoshop is not a computer animation application, it is one of the top of the line graphics programs. It is created by Adobe. Photoshop runs both on Macs and PC Windows, and even on SGIs. It can be used to touch up digitized images or to create graphics from scratch.

Adobe Premiere

Adobe Premier, just like the name says, is created by Adobe. It is a tool used to composite digitized video, stills, and apply a veriety of transitions and special effects. Adobe Premiere runs, both on Macintoshes and PCs Windows.

AliasIWavefront

Alias is one of the topmost computer animation packages out there. Alias was produced by the company that used to be Alias, but now it joined with Wavefront and is known as Alias. It runs on SGI's. Alias is well known for its great modeler which is capable of modeling some of the most complicated objects. Also, this software package is very flexible, allowing for programmers to create software that will run hand in hand with Alias.

Animator Studio

Animator Studio is a cell animation program from AutoDesk. Its predecessor was Animator Pro for PC DOS. Animator Studio runs under Windows. It has a multitude of features that minimize the animation creation time.

Elastic Reality

Elastic Reality is one of the top of the line morphing programs. Elastic Reality runs on Macs and SGIs. One of the great features of Elastic Reality as opposed to other programs is that, it uses splines as opposed to points to define the morphing area. Elastic Reality allows us to morph video as well as still images.

SoftImage

One of the most popular computer animation software packages. SoftImage is used in many top production studios around the country and around the world.

Strata Studio Pro

Strata Studio Pro is probably the most known 3D graphics application on the Mac. It is created by Strata Inc. Strata Studio Pro is mainly a still graphic rendering application, but it does have animation capabilities. Graphics for some games such as Myst were created in Strata Studio Pro.

1.6 APPLICATIONS FOR COMPUTER ANIMATION

Now-a-days, animation influences almost every aspect of life right from entertainment to education to security and many more areas. Here are some domains in which animation has, played a great role and many more applications and domains are about to come. Some applications in different domains are:

Entertainment: Games, Advertising, Film, Television, Video, Multimedia, are some of the entertainment fields in which computer animation has wide contribution, the topic is self explanatory as you all are exposed to the usage of animation in these fields from your day to day life programs on televisions, computers, etc.

Film: Computer animation has become regular and popular in special effects. Movies such as "*Jurassic Park*", "*Terminator 2: Judgment Day*", and "*The Abyss*" have brought computer animation to a new level in their films. Scale models are a fast and cost effective method of creating large alien scenes. But animation has done just as well in animating fire, smoke, humans, explosions, and heads made out of water.

A major part in integrating live film and the computer animation is to make absolutely sure that the scale and perspective of the animations are right. The scale is important to making the animation believable. The animators go through a great deal of work to make sure this is right. Usually computer animation is only used when the scene needed would be impossible or very difficult to create without it.

Television: Computer Animation plays a great role in television. Most of the titles on the television programs, news casts, and commercials, are done with computer animation. In the past, when computers were not a part of the process, animations were done with live video, cel animation, scale models, and character generators. Now, with the advent of computers, special programs could be used (i.e., computer painting, 3-D animation, motion control, and digital compositing programs).

Computer animation has simplified the making of television program titles, because of the versatility of computer generated animations, almost anything is possible. An animator can have a totally computer generated animation or have an animation with live video integrates, or even live video with animation integrated. Computer animation has advantaged the media also desires. With computer animation, professional animators can use pre-made templates to create animations for broadcasting within minutes of receiving the news.

Video: Everyone heard of animated cartoons. There is a new era of cartoons emerging on television. Computer animated cartoons can be produced much faster than cell animated ones. This is because, the animator does not have to draw every single frame, but only has to create a keyframe using, which the computer generates the in-between frames.

Sometimes computer animation looks more realistic. Sometimes, it is even possible to create computer animations that look realistic so that a person might not be able to tell, if it is real or not by simply looking at it, but this requires, enormous team effort.

Education: Now-a-days, studies of subjects like, Art, Physics, Chemistry, Maths, Biology, Medicine, Engineering, Technology, etc., are quite simple and interactive through the concept of E-Learning, where the electronic data like, educational CD's, websites, T.V. programs are contributing a lot. To make studies quite simple, animation plays an important role in fields that excessively need use animation for better understanding, are:

Physics: Say some students want to study the concept of rocket propulsion and its related aspects like, velocity, payload etc. Then, by using animation, we can easily describe the forces applicable at any instant, causing respective changes in different parameters of rocket. Without animation the teaching of concepts may not be that simplified because understanding from books doesn't include motion and understanding from video can't describe the applied forces and respective directions.

Mathematics: Probability, permutation, combination, etc., are some of the areas which can be well explained with the help of animation, which helps in enhancing the learning of the student.

***Chemistry**:* Computer animation is a very useful tool in chemistry. Many things in chemistry are too small to see, handle, or do experiments on, like, atoms and molecules for example, computer animation is the perfect tool for them. Chemistry teachers can create realistic models of molecules from the data they have and look at the way these molecules will interact with each other. They can get a full 3D picture of their experiments and look at it from different angles. Computer animation also allows them to do things that would be extremely hard to do in real life. For example, it is visible to construct models of molecules *on a computer.* People are always looking for new ways to educate their children. If, they are having fun, they learn better. Computer animation can be used to make very exciting and fun videos into which education can easily be incorporated. It is much more interesting to learn maths for example, when the letters are nice and colorful and flying around your TV screen instead of solving problems on plain black and white paper. Other subjects such as science, English, foreign languages, music, and art can also be tought by using computer animation.

***Engineering**:* CAD has always been an imperative tool in industry. For instance in automobile design, CAD could be used to model a car. But with the advent of computer animation, that model could now be changed into a full 3-D rendering. With this advantage, automobile makers could animate their moving parts and test them to make sure these parts don't interfere with anything else. This power helps car makers a lot by ensuring that the model of car will have no defects.

Art: Just like conventional animation, computer animation is also a form of art. A multitude of effects can be created on a computer than on a piece of paper. An artist can control a magnitude of things in a computer animation with a few clicks of a mouse than he can do in the conventional animation methods. A light source can be moved very easily, changing the way an entire scene looks. Textures can be changed just as easily, without redoing the whole animation. Computer graphics are not very likely to replace conventional methods anywhere in the future. There are still many things that cannot be done on the computer that an artist can do with a paintbrush and a palette. Computer graphics is simply just another form of art.

Advertising: One of the most popular uses for computer animation is in television advertising. Some of the models that the commercials would call for would be extremely difficult to animate in the past (i.e., Saxophones, boxes of detergent, bathrooms, kitchens, etc.) The modeled objects would then be animated, and incorporated with live video.

Archeology: With the advent of the computer, the archeologist has acquired a new tool, computer animation. A model of an object can be made relatively quickly and

without any wear and tear to the artifact itself using a 3D digitizer. All the scenery is modeled and put together into a single scene. Now, the archeologist has a complete model of the site in the computer. Many things can be done to this model. The exact position of artifacts is stored in the computer and can be visualized without visiting the excavation site again.

Architecture: One of the reasons for the development of virtual reality (which is actually a form of computer animation) was that it was going to be very useful to architects. Now, that has proved to be true. A person can hire an architect half way across the world over the Internet or other network. The architect can design a house, and create a walkthrough animation of the house. This will show the customer what the house will actually look like before anyone lays a hand on a hammer to build it.

Computer animation can also be helpful to architects so that they can see any flaws in their designs. This has proved to be very cost and time saving because no one has to build anything. This is one field, in which computer animation has proved to be extremely useful.

Military: In order to enter the military, one has to go through a lot of training. Depending on whether you want to be in the army, navy, or the marines, you might be working with equipment worth hundreds of thousands or even millions of dollars. The military wants to be sure you know, how to use this equipment before they actually let you use it. Training in simulators instead of on the battleground is proving to be a much cheaper and safer approach. Let us take the air force, for example, one has to learn how to fly a fighter jet.

Using computer animations in flight simulation is a very useful tool. Using animation a programmer can replicate real time flying. By creating a camera showing the view through the cockpit window, a pilot could fly through either virtual worlds or real animated places with all the natural disasters, and difficulties that could happen, if flying a real plane.

In this virtual world, the pilot would witness the usual distractions that a real pilot would, for instance, transport buses drive along the runway, and other planes take off and land. The programmer can put any type of weather condition or scenario into the animation.

Forensics: Accidents happen every minute. Very often, there are no witnesses except for the individuals involved in the accident or, worse yet, there are no surviving witnesses. Accident reconstruction is a field in which computer animation has been very useful. People arguing for it say that it offers, the ability for the court to witness the accident from more than just a bystander's perspective. Once, the reconstruction has been done, the camera can be placed anyway in a scene. The accident may be seen from either driver's perpective, or even birds eye view.

New animation systems allow detectives to recreate terrains and surroundings and actually calculate different things such as angles of bullet shots or levels of visisbility. This is extremely useful since very often, the site of an accident may have changed a lot since the time of the mishap.

Computer animation can also be used to simulate the landscape in which an operation will be going on. A satellite altitude picture can be converted into a 3D model using software and then animated with trees and under different weather.

Medicine: It is very hard for a doctor to get inside a living human body and to see what is happening. Computer animation once again comes in very useful. Every single organ in the body has already been modeled in a computer. A doctor, for example, can fly through these models and explore the area of the body s/he is going to be operating

on in order to get a better picture of the operation and possibly increase the success rate.

Another very important use of computer animation in medicine is to look at living tissue or organ of a patient and to explore it, and to see what if anything is wrong with it, without even making a single incision. Data can be gathered from a living specimen painlessly by means of various sensing equipment. For example, an MRI (Magnetic Resonance Imaging) scan takes pictures of across-sections of a part of a body (brain for example) every half a centimeter. Then, the data is transmitted to a computer, where a model is constructed and animated. A doctor can get a very clear picture of undisturbed tissue the way it looks in a body. This is very helpful in detecting abnormalities in very fragile parts of the body such as the brain.

With recent advances in the computer industry, people have developed faster and better computer hardware. Systems are underway which allow doctors to conduct operations with only a couple of small incisions through which instruments can be inserted. The use of virtual reality has allowed doctors to train on virtual patients using this procedure without once opening up a cadaver.

Multimedia*:* Multimedia is the use of various media to present a certain subject. This presentation itself is a multimedia presentation in the sense, it brings together graphics and text. Multimedia presentations can include text, graphics, sounds, movies, animations, charts, and graphs. Using computer animation in multimedia presentations is growing extremely popular since, they make a presentation look more professional and more pleasing to the eye. Computer animation is also very useful in demonstrating how different processes work.

Simulation*:* There are many things, places, and events people cannot witness in first person. There are many reasons for this. Some may happen too quickly, some may be too small, others may be too far away. Although people cannot see these things, data about them may be gathered using various types of sensing equipment. From this data models and simulations are made. Using computer animation for these simulations has proven very effective. If, enough data has been gathered and compiled correctly, a computer animation may yield much more information than, a physical model. One reason for this is that a computer animation can be easily modified and simply rendered (Rendering is the process a computer uses to create an image from a data file. Most 3D graphics programs are not capable of drawing the whole scene on the run with all the colours, textures, lights, and shading. Instead, the user handles a mesh which is a rough representation of an object. When the user is satisfied with the mesh, s/he then renders the image) to show changes. It is, not that easy however, to do this with a physical model. Another reason for using computer animation to simulate events as opposed to models is that variables can be programmed into a computer and then, very easily changed with a stroke of a button.

Space Exploration: As of now, the farthest point away from earth that the human was on is the moon, but we continually want to learn more. A trip by a human to another planet would take way too long. This is why we, have sent satellites, telescopes, and other spacecraft into space. All of these spacecraft continually send data back to earth. Now, all we have to worry about is presenting that data so it makes sense. This is where computer animation comes in. It can show an incredible amount of data visually, in the way that humans perceive it the best.

Much of the data sent from spacecraft can be input into a computer which will in turn generate an awesome looking animation so that one may actually navigate, explore, and see the distant worlds as if, we, were actually there.

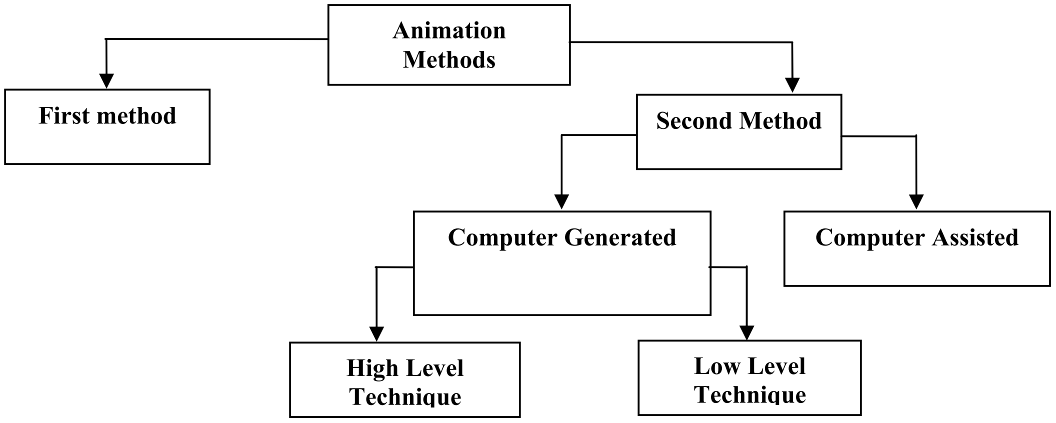
Computer animation can also be used to **design** satellites and other spacecraft more effitiently. Another possible use of computer animation is to plan the routes of future

ships to make sure there is nothing wrong with the path and so that a ship can gather the most data possible.

Note: The usage of computer animation is not restricted to only these fields, it is a creative process and has enormous applications in a wide range of fields.

1.7 SUMMARY

- **Traditional and historical methods for production of animation**



- **Definition:** Computer animation is a time based phenomenon of imparting visual changes in a scene according to any time sequence , the visual changes could be incorporated through positional changes , in object size , color , transparency , or surface texture etc.

- **Traditional Animation techniques :**
  1) Key Frames **2) Cel Animation**

Formula:
Required Key frames for a film

$$= \frac{\{[\text{Time(in seconds)}]*[\text{frames required per second(in gen.eral =24)}]\}}{\{\text{no. of in between frames}\}}$$

- **Types of Animation Systems : Keyframe, scripting, parameterized**

- **Morphing:** Transformation of object shapes from one form to another is called **morphing** (short form of metamorphism)-morphing methods can be applied to any motion or transition involving a change in shape.

- **Panning:** (i.e., shifting of background/foreground to give the illusion that the camera seems to follow the moving object, so that the background/ foreground seems to be in motion), etc.

- **Types of Animation :**

  Procedural Animation **Representational Animation**
  Stochastic Animation **Behavioural Animation**

- **Different ways of simulating motion:-**
  Zero Acceleration (Constant Speed) **Non-Zero Accelerations**
  Positive accelerations **Negative accelerations**

Combination of accelerations

- **Animation Tools:**

  Hardware tools: PCs ,Macintosh, Amiga
  Software tools

  Softimage (Microsoft) ; Alias/ Wavefront (SGI)
  3D studia MAX (Autodesk) ; Lightwave 3D (Newtek)
  Prism 3D Animation Software (Side Effects Software)
  HOUDINI (Side Effects Software)
  Apple's Toolkit for game developers ; Digimation etc.

- **Application of animation:** There are a variety of uses for computer animation. They can range from fun to practical and educational ones. Military, medicine, education, entertainment, etc., are some domains in which animation has played a great role and many more applications and domains are about to be opened up.

1.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) *Computer animation is a time based phenomenon of imparting visual changes to a scene according to any time sequence. The visual changes could be incorporated through positional changes, in object size, colour, transparency , or surface texture, etc.*

   Production of animation is done by two methods:

   First method is by artists creating a succession of cartoon frames, which are then combined into a film.

   Second method is by using physical models which are positioned to the image, where the image is recorded; then the model is moved to the next image for its recording, and this process is continued.

2) Two main categories of computer animation:

   a) *Computer-assisted animation which* usually refers to two dimensional systems that computerize the traditional animation process. Interpolation between key shapes is typically the only algorithmic use of the computer in the production of this type of animation.

   b) *computer generated animation*. is the animation presented via film or video. This is possible because the eye-brain assembles a sequence of images and interprets them as a continuous movement. Persistence of motion is created by presenting a sequence of still images at a fast enough rate to induce the sensation of continuous motion. Motion specification for computer-generated animation is divided into two categories: *Low level techniques* (techniques that aid the animator in precisely specifying motion) and *High level techniques* (techniques used to describe general motion behaviour).

3)

Low level techniques	High level techniques
Low level techniques provide aid to the animator in precisely specifying the motion. It involves techniques such as shape interpolation, algorithms which help the animator fill in the details of the motion. Here the animator usually has a fairly specific idea of the exact motion that he or she wants.	High level techniques used to describe general motion behavior. These techniques are algorithms or models used to generate a motion using a set of rules or constraints. The animator sets up the rules of the model, or chooses an appropriate algorithm, and selects initial values or boundary values. The system is then set into motion and the motion of the objects is controlled by the algorithm or model. This approach often relies on fairly sophisticated computation such as vector algebra and numerical techniques and others.
Cel Animation	**Key Frames**
When creating an animation using this method, each character is drawn on a separate piece of transparent paper. A background is also drawn on a separate piece of opaque paper. Then, when it comes to shooting the animation, the different characters are overlaid on top of the background in each frame. This method also saves time in that the artists do not have to draw in entire frames, but rather just the parts that need to change such as individual characters. Even separate parts of a character's body are placed on separate pieces of transparent paper	After a storyboard has been laid out, the senior artists go and draw the major frames of the animation. These major frames are frames in which a lot of change takes place. They are the key points of the animation. Later, a bunch of junior artists draw in the frames in between. This way, the workload is distributed and controlled by the key frames. By doing work this way, the time in which an animation can be produced is cut dramatically, depending on the number of people working on the project. Work can be done simultaneously by many people, thus cutting down on the time needed to get the final product out.

4) We cannot say which technique is better because different techniques are used in different situations. In fact, all these animation techniques are great, but they are most useful when they are all used together. Cel animation by itself would not help out much if it wasn't for key frames and being able to distribute the workload across many people.

Check Your Progress 2

1) Film duration= 30 seconds
   No. of frames required per second=24
   No. of frames required in entire film=24 * 30=720
   That is, we would need 720 frames for 30-second animation film;
   If the duplication of frames is allowed then the number of frames will be halved

2) Film duration = 30 seconds
   No. of frames required per second = 24
   No. of in-between frames = 3
   No. of frames required in entire film=(24 * 30)/3=240
   That is, we would need 240 frames for a half-minute animation film if number of in- between frames is 3.

3) Animation is an application based on the principle of persistence of vision that is $(1/16)^{th}$ of a second , so if we have approximately 24 frames change per second then our eye will not be able to identify the discontinuities in the animation scene .

If the number of frames decreases to less than 24 frames per second then possibility of detecting the discontinuities in the scene increases and our animation will not be effective

4) The sequence of steps to produce a full animation would be as follows:
   a) Develop a script or story for the animation.
   b) Lay out a storyboard, that is a sequence of informal drawings that shows the form, structure, and story of the animation.
   c) Record a soundtrack.
   d) Produce a detailed layout of the action.
   e) Correlate the layout with the soundtrack.
   f) Create the "key frames" of the animation. The key frames are those where the entities to be animated are in positions such that intermediate positions can be easily inferred.
   g) Fill in the intermediate frames (called "in-betweening" or "tweening").
   h) Make a trial "film" called a "pencil test".
   i) Transfer the pencil test frames to sheets of acetate film, called "cels". These may have multiple planes, e.g., a static background with an animated foreground.
   j) The cels are then assembled into a sequence and filmed

Check Your Progress 3

1) Whenever we require to have some realistic display in many applications of computer animation like, accurate representation of the shapes of sea waves, thunderstorm or other natural phenomenon, which can be described with some numerical model, the accurate representation of the realistic display of scene measures the reliability of the model. Computer Graphics are used to create realistic elements which are intermixed with live action to produce animation. But in many fields realism is not the goal, like physical quantities are often displayed with pseudo colours or abstract shapes that change over time.

2) Frame animations is an "internal" animation method, i.e., it is an animation inside a rectangular frame where a sequence of frames follow each other at a fast rate, fast enough to convey fluent motion. And it is best suited for cartoon movies. Sprite animation is an interactive – external animation where the animated object interaction script is written by the programmer, every time an animob touches another animob or when an animob gets clicked, the script is activated and decides what is to be done. These features are usefull in the gaming systems.

3) Animated objects (sprites or movies) are refered as "animobs", which are used in the gaming applications designed using sprite animation. That is these are programmable animated objects , which can respond to the interactive environment according to the scripts written by the programmers.

4) Morphing is short form of metamorphism which means transformation of object shapes from one form to another. Morphing methods can be applied to any motion or transition involving a change in shape. Panning means shifting of background/foreground to give the illusion of camera in motion following a moving object, so that the background/ foreground seems to be in motion. Both techniques are widely used in animation application.

Check Your Progress 4

1) a) If the distance between frames is constant then the motion will neither be accelerated nor decelerated. In fact the uniform motion will be simulated in the animation.

b) If the Distance between frames continuously increases, then accelerated motion will be simulated in the animation.

c) If the distance between frames continuously decreases, then decelerated motion will be simulated in the animation.

2) Uniform motion will be simulated by the straight line. As straight line leads to a constant distance between frames, the motion will neither be accelerated nor decelerated. In fact the uniform motion will be simulated in the animation.

3) By definition computer animation is a time-based phenomenon of imparting visual changes in a scene according to any time sequence. The visual changes could be incorporated through positional changes, in object size, color, transparency, or surface texture, etc. So, if the spacing between frames is more, then it means that less frames appear in the same duration (in-between frames are less). Thus there are fast positional changes in the images drawn on the frames, imparting to fast motion simulation in the animation. For the same reason, the decreasing spacing between frames contributes to simulate deceleration.

UNIT 2 MULTIMEDIA

Structure **Page Nos.**

2.0 Introduction 32
2.1 Objectives 33
2.2 Concept of Hypertext and Hypermedia 33
2.2.1 Definition of Hypertext 34
2.2.2 Definition of Hypermedia 34
2.2.3 Understanding the Concept 34
2.2.4 Hypertext/media and Human Memory 35
2.2.5 Linking 36
2.3 Multimedia Application 37
2.3.1 What is Multimedia 37
2.3.2 Importance of Multimedia 38
2.3.3 Role in Education and Training 38
2.3.4 Multimedia Entertainment 40
2.3.5 Multimedia Business 41
2.3.6 Video Conferencing and Virtual Reality 41
2.3.7 Electronic Encyclopedia 42
2.4 Graphics 42
2.4.1 What is Graphics 42
2.4.2 Types of Graphic Images 44
2.4.3 Graphic Files Compression Formats 47
2.4.4 Uses for GIF and JPEG Files 51
2.5 Audio and Video 53
2.5.1 Sound and Audio 53
2.5.2 Analog Sound Vs Digital Sound 53
2.5.3 Audio File Formats 56
2.5.4 Image Capture Formats 57
2.5.5 Digital Video 59
2.5.6 Need for Video Compression 62
2.5.7 Video File Formats 62
2.6 Multimedia Tools 64
2.6.1 Basic Tools 64
2.6.2 Types of Basic Tools 65
2.6.3 Authoring Tools 67
2.6.4 Types of Authoring Tools 68
2.6.5 Multimedia Tool Features 69
2.7 Summary 70
2.8 Solutions/Answers 70
2.9 Further Readings 75

2.0 INTRODUCTION

Multimedia is a new aspect of literacy that is being recognised as technology expands the way people communicate. The concept of literacy increasignly, is a measure of the ability to read and write. In the modern context, the word, means reading and writing at a level adequate for written communication. A more fundamental meaning is now needed to cope with the numerous media in use, perhaps meaning a level that enables one to function successfully at a certain status in society. Multimedia is the use of several different media to convey information. Several different media are already a part of the canon of global communication and publication: (text, audio, graphics, animation, video, and interactivity). Others, such as virtual reality, computer programming and robotics are possible candidates for future inclusion. With the widespread use of computers, the basic literacy of

'reading' and 'writing' are often done via a computer, providing a foundation stone for more advanced levels of multimedia literacy.

Multimedia is the use of several media (e.g. text, audio, graphics, animation, video) to convey information. Multimedia also refers to the use of computer technology to create, store, and experience multimedia content.

In this unit, we will learn about the basics of multimedia and its applications including graphics, audio, video etc. We will also learn some basic multimedia authoring tools.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- describe hypertext and hypermedia concepts,
- describe how multimedia applications are influencing every aspect of life,
- discuss different file formats used for multimedia applications, and
- give basic description of various multimedia tools.

2.2 CONCEPT OF HYPER TEXT AND HYPER MEDIA

Any student, who has used online help for gaming etc., will already be familiar with a fundamental component of the Web-Hypertext.

Hypertext is the concept whereby, instead of reading a text in a liner fashion (like a book), you can at many points jump from one place to another, go forward or back, get much more detail on the current topic, change direction and navigate as per your desire.

- **Hypertext:** Hypertext is conceptually the same as regular text - it can be stored, read, searched, or edited - with an important difference: hypertext is text with pointers to other text. The browsers let you deal with the pointers in a transparent way -- select the pointer, and you are presented with the text that is pointed at.

- **Hypermedia:** Hypermedia is a superset of hypertext. Hypermedia documents contain links not only to other pieces of text, but also to other forms of media - sounds, images, and movies. Images themselves can be selected to link to sounds or documents. Hypermedia simply combines hypertext and multimedia.

Some examples of Hypermedia might be:

- You are reading a text that is written in Hindi. You select a Hindi phrase, then hear the phrase as spoken in the native tongue.
- You are viewing a manufacturing plant's floor plan, you select a section by clicking on a room. The employee's name and picture appears with a list of their current projects.
- You are a law student studying the University Revised Statutes. By selecting a passage, you find precedents from a 1920 Supreme Court ruling stored at Law Faculty. Cross-referenced hyperlinks allow you to view any one of 500 related cases with audio annotations.

Hypertext and HyperMedia are concepts, not products and both terms were coined by **Ted Nelson**.

2.2.1 Definitions of Hypertext

- A way of presenting information online with connections between one piece of information and another. These connections are called hypertext links. Thousands of these hypertext links enable you to explore additional or related information throughout the online documentation. See also hypertext link.
- This term describes the system that allows documents to be cross- linked in such a way that the reader can explore related documents by clicking on a highlighted word or symbol.
- A non-sequential method for reading a document displayed on a computer screen. Instead of reading the document in sequence from beginning to end, the reader can skip to topics by choosing a highlighted word or phrase embedded within the document. This activates a link, connecting the reader to another place in the same document or to another document. The resultant matrix of links is called a web.
- This is a mark-up language that allows for non-linear transfers of data. The method allows your computer to provide the computational power rather than attaching to a mainframe and waiting for it to do the work for you.
- In computing, hypertext is a user interface paradigm for displaying documents which, according to an early definition (Nelson 1970), "branch or perform on request." The most frequently discussed form of hypertext document contains automated cross-references to other documents called hyperlinks. Selecting a hyperlink causes the computer to display the linked document within a very short period of time.

2.2.2 Definitions of Hypermedia

- **Hypermedia** is a term created by **Ted Nelson** in 1970. It used as a logical extension of the term hypertext, in which graphics, audio, video, plain text and hyperlinks intertwine to create a generally non-linear medium of information. This contrasts with multimedia, which, although often capable of random access in terms of the physical medium, is essentially linear in nature. The difference should also be noted with hypergraphics or super-writing which is a Lettrist form from the 1950s which systemises creativity across disciplines.

A classic example of hypermedia is World Wide Web, whereas, a movie on a CD or DVD is an example of standard multimedia. The difference between the two can (and often do) blur depending on how a particular technological medium is implemented. The first hypermedia system was the Aspen Movie Map.

2.2.3 Understanding the Concept

For understanding the concept of Hypertext and Hypermedia we will look at how the human memory works.

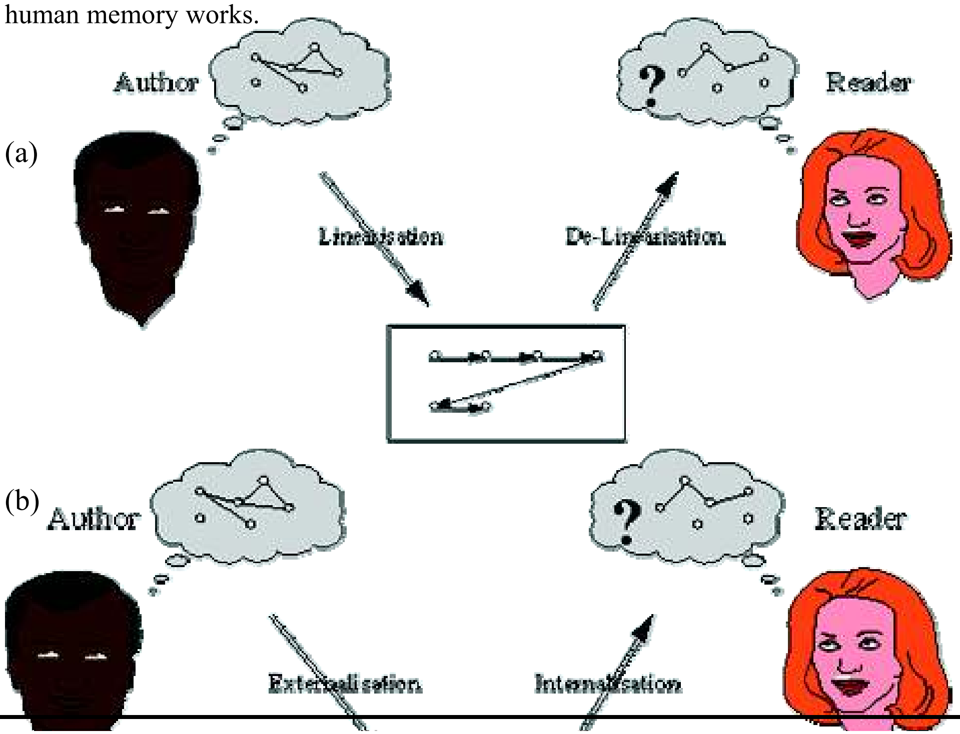


Figure 1: (a) Process of writing and reading using traditional linear media (b) Process of writing and reading using non-linear hypermedia.

2.2.4 Hypertext/media and Human Memory

Humans associate pieces of information with other information and create complex knowledge structures. Hence, it is also said that the human memory is associative. We often remember information via association. For example, a person starts with an idea which reminds of a related idea or a concept which in turn reminds him/her of another idea. The order in which a human associates an idea with another idea depends on the context under which the person wants information.

When writing, an author converts his/her knowledge which exists as a complex knowledge structure into an external representation. Information can be represented only in a linear manner using physical media such as printed material and video tapes. Therefore, the author has to convert his/her knowledge into a linear representation using a linearisation process. This is not easy. So the author will provide additional information, such as a table of contents and an index, to help the reader understand the overall organisation information.

The reading process can be viewed as a transformation of external information into an internal knowledge base combined with integration into existing knowledge structures, basically a reverse operation of the writing process. For this, the reader breaks the information into smaller pieces and rearranges those based on the readers' information requirement. We rarely read a text book or a scientific paper from start to finish. We tend to browse through the information and then follow the information headings that are interesting to us.

Hypermedia, using computer enabled links, allows us to partially imitate writing and reading processes as they take place inside our brain. We can create non linear information structures by associating pieces of information in different ways using links. Further, we can use a combination of media comprising of text, images, video, sound and animation for value addition in the representation of information. It is not necessary for an author to go through a linearisation process of his/her knowledge when writing. Also, the reader can access some of the information structures the author had when writing the information. This will help the reader create his/her own representation of knowledge and to amalgamate that knowledge into the existing knowledge structures.

In addition to being able to access information through association, hypermedia applications are supported by a number of additional aspects. These include an ability to incorporate various media, interactivity, vast data sources, distributed data sources, and powerful search engines. All these make hypermedia an extremely powerful tool to create, store, access and manipulate information.

2.2.5 Linking

Hypermedia systems as well as information in general contains various types of relationships between various information elements. Examples of typical relationships include similarity in meaning or context , similarity in logical sequence or temporal sequence, and containment.

Hypermedia allows these relationships to be installed as links which connect the various information elements, so that these links can be used to navigate within the information space.

One possible structure is based on the mechanics of the links. We can also look at the number of sources and destinations for links (single-source single-destination, multiple-source single-destination, etc.) the directionality of links (unidirectional, bi-directional), and the anchoring mechanism (generic links, dynamic links, etc.).

A more useful link structure is based on the type of information relationships being represented. In particular, we can divide relationships into those based on the organisation of the information space called structural links and those related to the content of the information space called associative and referential links.

Let us take a brief look at these links.

Structural Links: The information contained within the hypermedia application is typically organised in some suitable fashion. This organisation is represented using structural links. We can group structural links together to create different types of application structures. If we look, for example, at a typical book, then this has both a linear structure i.e. from the beginning of the book linearly to the end of the book and usually a hierarchical structure in the form of the book contains chapters, the chapters contain sections, the sections containing matter. Typically in a hypermedia application we try to create and utilise appropriate structures.

Associative Links: An associative link is a link which is completely independent of the specific structure of the information. For instance we have links based on the meaning of different information components. The most common example which most people would be familiar with is cross-referencing within books for example – for more information on X refer to Y. It is these relationships - or rather the links which are a representation of the relationships – which provide the essence of hypermedia, and in many respects can be considered to be the defining characteristic of hypermedia.

Referential Links: A third type of link is a referential link. It is related to the associative link. Rather than representing an association between two related concepts, a referential link provides a link between an item of information and an explanation of that information. A simple example would be a link from a word to a definition of that word. One simple way of understanding the difference between associative and referential links is that the items linked by an associative link can exist independently, but are related at a conceptual level.

☞Check Your Progress 1

1) Define hypertext and hypermedia?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

………

2) Explain the concept of hypermedia/text in terms of human memory.

……………………………………………………………………………………

……………………………………………………………………………………

………………………………………………………………………………………
………

3) Illustrate various links used in hypermedia.

   ………………………………………………………………………………………
   ………………………………………………………………………………………
   ………………………………………………………………………………………
   ………

2.3 MULTIMEDIA APPLICATIONS

Multimedia, the term itself clarifies that, it is a combination of different medias of communication like, text, graphic, audio etc. Now-a-days this field of multimedia is taken as the tool as well as one of the best option to communicate your throughout electronically.

In the section, after having briefings of the descipline of multimedia we will discuss its application in various fields.

2.3.1 What is Multimedia

Introduction

People only remember 20% of what they see and 30% of what they hear. But they remember 50% of what they see and hear, and as much as 80% of what they see, hear, and do simultaneously. Computer Technology Research, 1993

Multimedia is any mixture of text, graphics, art, sound, animation and video with links and tools that let the person navigate, interact, and communicate with the computer. When you allow the viewer to control what and when these elements are delivered, it is **interactive multimedia.** When you provide a structure of linked elements through which the learner can navigate, interactive multimedia becomes **hypermedia.**

Although the definition of multimedia is simple, making it work can be very complex. Not only do you need to understand how to make each multimedia element work, but you also need to know how to effectively blend the elements together using educational multimedia computer tools. If done properly, **interactive multimedia** excels in leaving lasting impressions in the learning process. Retention rates increase by 25% to 50%.

Interactive Multimedia: What is "interactive", "multi" and "media" about it?

Interactive: Users can use a variety of input devices to interact with the computer, such as a joystick, keyboard, touch screen, mouse, trackball, microphone, etc.

Multi refers to the multiple file usages used in the multimedia product, such as sound, animation, graphics, video, and text.

Media: Many media sources can be used as components in the multimedia product, such as a videodisk, CDROM, videotape, scanner, CD or other audio source, camcorder, digital camera, etc. Media may also refer to the storage medium used to store the interactive multimedia product, such as a videodisk or CDROM.

Examples of environments where interactive multimedia is being used

- Touch screen kiosks (museums, hospitals, bank lobbies)
- Distance education (via computer, compressed video, satellite...)
- Interactive, educational software on CDROM or videodisk
- Virtual Reality "theatres".

2.3.2 Importance of Multimedia

Multimedia will help spread the information age to millions of teachers/learners. Multimedia educational computing is one of the fastest growing markets in the world today.

Multimedia is fast emerging as a basic skill that will be as important to life in the twenty-first century as reading is now. In fact, multimedia is changing the way people read, interact and distribute information. Instead of limiting one to the linear re-presentation of text as printed in books, multimedia makes reading enjoyable with a whole new dimension by giving words an important new dynamics. In addition to conveying meaning, words in multimedia serve as triggers that readers can use to expand the text in order to learn more about a topic. This is accomplished not only by providing more text but by bringing it to life with audio, video and graphics.

Accelerating this growth are advances in technology and price wars that have dramatically reduced the cost of multimedia computers. The growing number of internet users has created a huge market for multimedia. The new tools are enabling educators to become developers. Noting how multimedia is used to enable individuals to create course material, that once required teams of specialists, individuals can now produce multimedia desktop video productions.

2.3.3 Role in Education and Training

Multimedia presentations are a great way to introduce new concepts or explain a new technology. Individuals find it easy to understand and use.
Multimedia can be used for education, training, simulations, digital publications, museum exhibits and so much more. With the advent of multimedia authoring applications like Flash, Shockwave and Director amongst a host of other equally enchanting applications are available in the market today. Your application of multimedia is only limited by your imagination. Training or instructional methods and advancement in technologies have always gone hand in hand. For example:

Historical method – Oral tradition:

The teacher was the only source of information
The teacher served as a role model.
The teacher was the primary resource to meet individual learning needs.

Printing Press discovered in 16th century:

Books provided more role models and multiple perspectives.
Exposure to books demanded that learners use critical thinking to resolve conflicting interpretations.
Teachers helped learners identify books, develop critical thinking skills, interpret different texts etc.
Books made learners independent of teacher. They had access to information which they could themselves read and learn on their own.

Photo and Video were discovered in the 19th century:

Visuals as add on to texts in books.
They enabled distance education.
They improved learning where verbal description was not adequate.

Teachers could select print, photo, video or some other combination to best suit teaching content.

Digital and Interactive Media has been developed in 20th century:

New media enhances visual and verbal content.
It doesn¹t replace earlier media.
New media allows dynamic alteration of instruction based on learner responses.
The teacher's role now is one of a guide and is not center stage any more.
Active learners create, integrate ideas, approach learning according to their interests and learning styles.

Use of Interactive Multimedia in Education

- Virtual reality, where 3-D experimental training can simulate real situations.
- Computer simulations of things too dangerous, expensive, offensive, or time-sensitive to experience directly Interactive tutorials that teach content by selecting appropriate sequencing of material based on the ongoing entry of student responses, while keeping track of student performance.
- Electronic presentations.
- Instruction or resources provided on the Internet (World Wide Web; 24 hours a day).
- Exploratory hypertext software (i.e. encyclopedias, databases) used for independent exploration by learners to complete research for a paper, project, or product development. They may use IMM resources to collect information on the topic or use multimedia components to create a product that blends visual, audio or textual information for effectively communicating a message.

Education courses, skills, and knowledge are sometimes taught out of context due to lack of application of real time examples. To overcome this, educators are using multimedia to bring into their classrooms real-world examples to provide a in-context framework important for learning. Multimedia and tools like the Internet give Faculty instant access to millions of resources.

Examples

- CyberMath
  - Animation, Plug-in,VRML (3D)
- Discovery Channel On-Line
  - Latest and greatest about the world we live in
- Frog Dissection
  - mpeg
- Dare Ware
  - Multimedia education software, "Talking Teacher
- Yahooligans
  - Answers to questions via e-mail
  - Several topics explained with complete class notes
- Scientific American
  - Easy to understand science, Good presentation
- National Geographic
  - Good multimedia - RealAudio, Chat

Education training procedures fall into three general categories:

1) **Instructor Support Products:** These are used by teachers in addition to text books, lectures and other activities within a class room environment.
2) **Standalone or Self Paced Products:** These are also called Computer based training and are designed for students to replace the teacher.

3) **Combination Products:** As the name implies these fall between support and standalone products. These are used by students on the directions of the instructors or to enhance classroom activities.

Education and training systems are built with three main objectives:

a) The learning objectives and purpose of the training.

b) Assessment or testing of the students to make sure they have learnt something.

c) The background and abilities of the student.

2.3.4 Multimedia Entertainment

The field of entertainment uses multimedia extensively. One of the earliest and the most popular applications of multimedia is for games. Multimedia made possible innovative and interactive games that greatly enhanced the learning experience. Games could come alive with sounds and animated graphics. These applications attracted even those to computers, who, otherwise would never have used them for any other application.

Games and entertainment products may be accessed on standard computer workstations via CDs or networks or on special purpose Game machines that connect to television monitors for display. These functions are quiet complex and challenging for the users.

These products rely on fairly simple navigational controls to enable the user to participate. Joystick and track ball are often used for moving objects, pointing guns or flying aircrafts while mouse buttons and keystrokes are used to trigger events like firing guns / missiles..

Multimedia based entertainment and game products depend on the use of graphics, audio, animation and video to enhance their operation. A game may include computer graphics taking the user on a hunt on a deserted island for hidden treasures or a princess. Audio is used for sound effects while video and animation are used for special effects.

These type of products also offer multi player features in which competition is managed between two or more players.

2.3.5 Multimedia Business

Even basic office applications like a MS word processing package or a MS Excel spreadsheet tool becomes a powerful tool with the aid of multimedia business.

Pictures, animation and sound can be added to these applications, emphasizing important points in the documents and other business presentations.

2.3.6 Video Conferencing and Virtual Reality

Virtual reality is a truly fascinating multimedia application. In this, the computer creates an artificial environment using hardware and software. It is presented to the user in such a way that it appears and feels real. Three of the five senses are controlled by the computer in virtual reality systems. Virtual reality systems require extremely expensive hardware and software and are confined mostly to research laboratories.

Another multimedia application is videoconferencing. When a conference is conducted between two or more participants at different sites by using computer networks to transmit audio and video data, then it is called video conferencing. A videoconference is a set of interactive telecommunication technologies which allow

two or more locations to interact via two-way video and audio transmissions simultaneously. It has also been called visual collaboration and is a type of groupware.

Digital compression of audio and video streams in real time is the core technology behind video conferencing. Codec is the hardware or software that performs compression . Compression rates of up to 1:500 can be achieved. The resulting digital stream of 1's and 0's is subdivided into labelled packets, which are then transmitted through a digital network usually ISDN or IP.

The other components required for a VTC(Video Tele Conference) system include:

Video input: video camera or webcam

Video output: computer monitor or television

Audio input: microphones

Audio output: usually loudspeakers associated with the display device or telephone

Data transfer: analog or digital telephone network, LAN or Internet

There are basically two kinds of VTC systems:

1) **Desktop systems** are add-ons to normal PC's, transforming them into VTC devices. A range of different cameras and microphones can be used with the board, which contains the necessary codec and transmission interfaces.
2) **Dedicated systems** have all required components packaged into a single piece of equipment, usually a console with a high quality remote controlled video camera. These cameras can be controlled from a distance to move left and right, tilt up and down, and zoom. They are known as PTZ cameras. The console contains all electrical interfaces, the control computer, and the software or hardware-based codec. Omnidirectional microphones are connected to the console, as well as a TV monitor with loudspeakers and/or a video projector.

There are several types of dedicated VTC devices.

Large group VTC are non-portable, large, more expensive devices used for large rooms and auditoriums.

Small group VTC are non-portable or portable, smaller, less expensive devices used for small meeting rooms. Individual VTC are usually portable devices, meant for single users, have fixed cameras, microphones and loudspeakers integrated into the console.

2.3.7 Electronic Encyclopedia

It is the application of multimedia for the creation of an encyclopedia with millions of entries and hypertext cross references covering a wide variety of research and refernece topics mainly for educational and training purposes.

☞Check Your Progress 2

1) What is interactive multimedia ?

    ……………………………………………………………………………………
    ……………………………………………………………………………………
    ……………………………………………………………………………………

……………………………………………………………………………………

…………

2) Explain the application of multiledia in education and training.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

…………

3) How does a video tele conference system work?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

…………

2.4 GRAPHICS

Graphics is one of the core component of any multimedia application. We all have heard a famous saying that "one picture conveys a message of 1000 words", so without graphics the multimedia is quite expressionless. So let us discuss the topic of graphics from multimedia point of view.

2.4.1 What is Graphics

Graphics is a term, which refers to any computer device or program that makes a computer capable of displaying and manipulating pictures. The term also refers to the images themselves.

For example, laser printers and plotters are graphics devices because they permit the computer to output pictures.

A graphics monitor is a display monitor that can display pictures.

A graphics board or card is a printed circuit board of which, when installed in a computer, permits the computer to display pictures.

Many software applications include graphics components. Such programs are said to support graphics. For example, certain word processors support graphics because they let you draw or import pictures. All CAD/CAM systems support graphics.

The following are also considered graphics applications :

- **Paint Programs:** Allow you to create rough freehand drawings. The images are stored as bit maps and can easily be edited.
- **Illustration/Design Programs:** Supports more advanced features than paint programs, particularly for drawing curved lines. The images are usually stored in vector-based formats. Illustration/design programs are often called draw programs.
- **Presentation Graphics** Software**:** This software lets you create bar charts, pie-charts, graphics, and other types of images for slide shows and reports. The charts can be based on data imported from spreadsheet applications.
- **Animation** Software**:** Enables you to chain and sequence a series of images to simulate movement. Each image is like a frame in a movie.
- **CAD Software:** Enables architects and engineers to draft designs.

- **Desktop Publishing:** Provides a full set of word-processing features as well as fine control over placement of text and graphics, so that you can create newsletters, advertisements, books, and other types of documents.

In general, applications that support graphics require a powerful CPU and a large amount of memory. Many graphics applications—for example, computer animation systems—require more computing power and hence, run only on powerful workstations or specially designed graphics computers. The same is also true of complex 3-D graphics applications.

In addition to the CPU and memory, graphics software requires a graphic monitor and support for one of the many graphics standards. Most PC programs, for instance, require VGA graphics. Sometimes this is inbuilt and sometimes it is an add on feature.

The quality of most graphics devices is determined by their resolution—how many points per square inch they can represent—and their colour capabilities.

Images have high information content, both in terms of information theory (i.e., the number of **bits** required to represent images) and in terms of the meaning that images can convey to the viewer. Because of the importance of images in any domain in which complex information is displayed or manipulated, and also because of the high expectations that consumers have of image quality, computer graphics have always placed heavy demands on computer hardware and software.

In the 1960s early computer graphics systems used vector graphics to construct images out of straight line segments, which were combined for display on specialised computer video monitors. Vector graphics is economical in its use of memory, as an entire line segment is specified simply by the coordinates of its endpoints. However, it is inappropriate for highly realistic images, since most images have at least some curved edges, and using all straight lines to draw curved objects results in a noticeable “stair-step” effect.

In the late 1970s and '80s raster graphics, derived from television technology, became more common, though was still limited to expensive graphics **workstation computers**. Raster graphics represents images by “**bit maps**” stored in the computer’s memory and displayed on a screen composed of tiny **pixels**. Each pixel is represented by one or more memory bits. One bit per pixel suffices for black-and-white images, while four bits per pixel specify a 16-step gray-scale image. Eight bits per pixel specifies an image with 256 colour levels; the so-called “true color” requires 24 bits per pixel (specifying more than 16 million colours). At that resolution, or bit depth, a full-screen image requires several megabytes (millions of bytes; 8 bits = 1 byte) of memory. Since the 1990s, raster graphics has become ubiquitous, personal computers are now commonly equipped with dedicated video memory for holding high-resolution bit maps.

2.4.2 Types of Graphic Images

Graphic images that have been processed by a computer can usually be divided into two distinct categories. Such images are either bitmap files or vector graphics

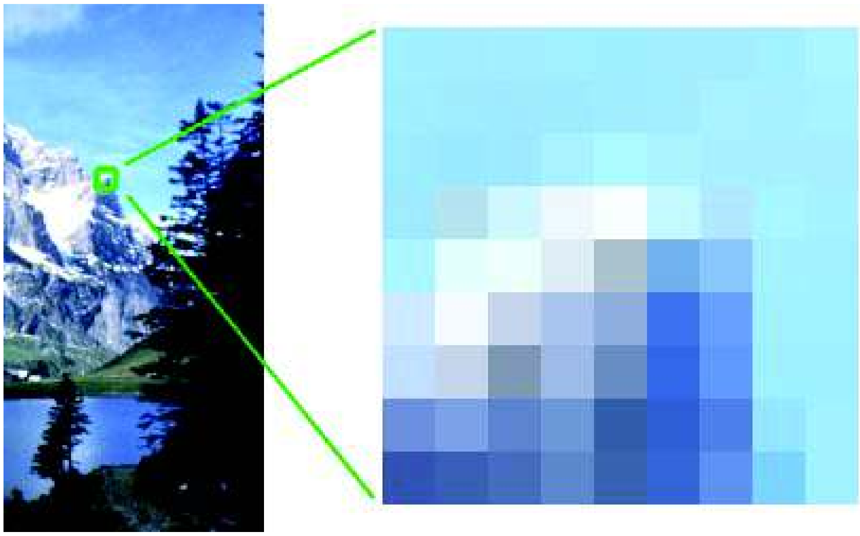
As a general rule, scanned images are bitmap files while drawings made in applications like Corel Draw or Illustrator are saved as vector graphics. But images between these two data types can be converted and it is even possible to mix them in a file.

Bitmap Graphics

The information below describes bitmap data.

Bitmap images are a collection of bits that form an image. The image consists of a matrix of individual dots (or pixels) that have their own colour described using bits.

Lets take a look at a typical bitmap image to demonstrate the principle:



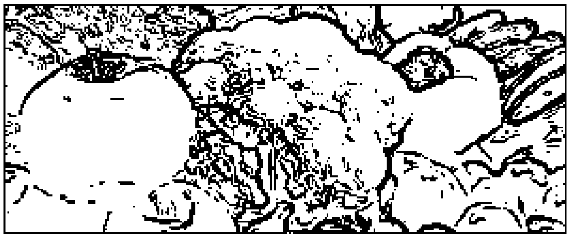
To the left you see an image and to the right a 250 percent enlargement of the top of one of the mountains. As you can see, the image consists of hundreds of rows and columns of small elements that all have their own colour. One such element is called a pixel. The human eye is not capable of seeing each individual pixel so we perceive a picture with smooth gradations.

Application of the image decides the number of pixels you need to get a realistic looking image.

Types of Bitmap Images

Bitmap images can contain any number of colours but we distinguish between four main categories:

1) Line-art: These are images that contain only two colours, usually black and white.



2) Grayscale images, which contain various shades of grey as well as pure black and white.

3) Multitones: Such images contain shades of two or more colours.

4) Full colour images: The colour information can be described using a number of colour spaces: RGB, CMYK for instance.

Characteristics of Bitmap Data

Bitmap data can take up a lot of room. A CMYK A4-size picture that is optimised for medium quality printing (150 lpi) takes up 40 MB. Compression can reduce the size of the file.

The image with the enlargement showed one of the main disadvantages of bitmap images: once they are enlarged too much, they look unnatural and blocky. But reducing a picture too much also has a bad influence as it looses sharpness.

Applications that can Handle Bitmap Data

There are hundreds of applications on the market that can be used to create or modify bitmap data. For example, Adobe PhotoShop, Corel Photo-Paint etc

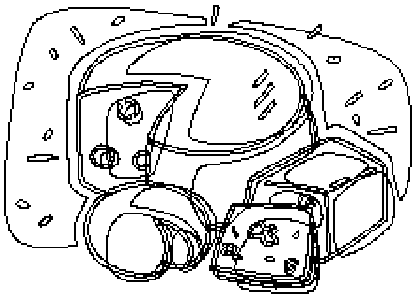
File Formats that are used for Bitmap Data

Bitmap data can be saved in a wide variety of file formats. Among these are:

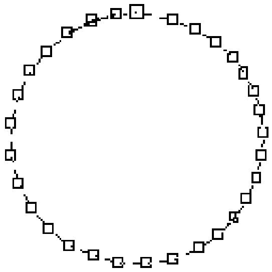
- BMP: limited file format that is not suitable for use in prepress.
- EPS: flexible file format that can contain both bitmap and vector data.
- GIF: mainly used for internet graphics.
- JPEG: or rather the JFIF file format, which is mainly used for internet graphics.
- PDF: versatile file format that can contain just about any type of data including complete pages, not yet widely used to exchange just images
- PICT: file format that can contain both bitmap and vector data but that is mainly used on Macintosh computers and is not very suitable for prepress.
- TIFF: the most popular bitmap file format in prepress

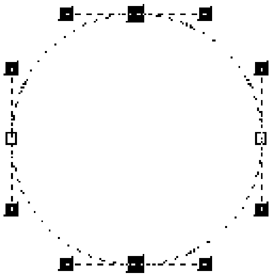
Vector Graphics

Vector graphics are images that may be entirely described using mathematical definitions. The image below shows the principle. To the left you see the image itself and to the right you see the actual lines that make up the drawing.



Each individual line is made up a large number of small lines that interconnect a large number of or, just a few control points that are connected using Bezier curves. It is this latter method that generates the best results and that is used by most drawing programs.





This drawing demonstrates the two principles. To the left a circle is formed by connecting a number of points using straight lines. To the right, you see the same circle that is now drawn using 4 points (nodes) only.

Characteristics of vector drawings

Vector drawings are usually pretty small files because they contain only data about the bezier curves that form the drawing. The EPS-file format that is often used to store vector drawings includes a bitmap preview image along the Bezier data.

The file size of this preview image is usually larger than the actual bezier data themselves.

Vector drawings can usually be scaled without any loss in quality. This makes them ideal for company logo's, maps or other objects that have to be resized frequently.

Applications that can Handle Vector Data

There are hundreds of applications on the market that can be used to create or modify vector data. In prepress, Adobe Illustrator, Corel Draw and Macromedia Freehand are the most popular.

File Formats that are used for Vector Data

This data can be saved in a wide variety of file formats. Among these are:

- EPS: the most popular file format to exchange vector drawings although EPS-files can also contain bitmap data.
- PDF: versatile file format that can contain just about any type of data including complete pages, not yet widely used to exchange just images.
- PICT: file format that can contain both bitmap and vector data but that is mainly used on Macintosh computers.

It is often necessary to convert images from bitmap data to vector data or back. Some possible uses include:

- Vector drawings often have to be converted to bitmaps if they will be used on a web page.
- If you scan a logo, it is a bitmap image but if it is going to be resized time and again depending upon its application then, it becomes more practical to have that logo as a vector drawing so its file size is smaller and you can change the size without worrying about any loss in quality.
- Vector drawings are sometimes too complicated for a RIP to be output on film or plate. Sometimes converting them to bitmap simplifies the file.

2.4.3 Graphic File Compression Formats

Web graphics are by necessity compressed because of the bandwidth issues surrounding networked delivery of information and because image files contain so much information. File format is the specific format in which the image is saved. The format is identified by the three-letter extension at the end of the file name. Every format has its own characteristics, advantages and disadvantages. By defining the file format it may be possible to determine the number of pixels and additional information. Each file format will have a reference to the numbers of bits per pixel that the format is capable of supporting

- 1 bit per pixel refers to an image with 2 colours
- 4 bit per pixel refers to an image with up to 16 colours
- Similarly 24 bits per pixel refers to 16,777,216 colours

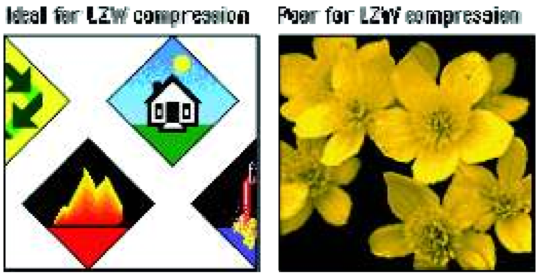
Different graphic file formats employ varying compression schemes, and some are designed to work better than others for certain types of graphics. The two primary Web file formats are GIF and JPEG.

Graphic Interchange Format (GIF)

The Graphic Interchange Format is an efficient means to transmit images across data networks. In the early 1990s the original designers of the World Wide Web adopted GIF for its efficiency and widespread familiarity. The overwhelming majority of images on the Web are now in GIF format, and virtually all Web browsers that support graphics can display GIF files. GIF files incorporate a compression scheme to keep file sizes at a minimum, and they are limited to 8-bit (256 or fewer colours) colour palettes.

GIF File Compression

The GIF file format uses a relatively basic form of file compression that squeezes out inefficiencies in the data storage without losing data or distorting the image. The LZW compression scheme is best at compressing images with large fields of homogeneous colour. LZW is the compression scheme used in GIF format. It is less efficient at compressing complicated pictures with many colours and complex textures as illustrated below with the help of two graphics.



Improving GIF Compression

Characteristics of LZW compression can be used to improve its efficiency and thereby reduce the size of your GIF graphics. The strategy is to reduce the number of colours in your GIF image to the minimum number necessary and to remove stray colours that are not required to represent the image. A GIF graphic cannot have more than 256 colors but it can have fewer colours, down to a minimum of two (black and white). Images with fewer colours will compress more efficiently under LZW compression.

Full color image dithered to 256 colors

Full color image dithered to 64 colors

Interlaced GIF

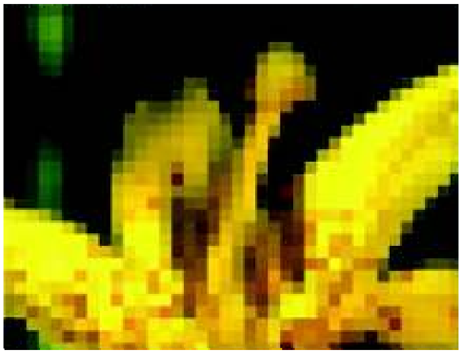
The conventional i.e. non-interlaced GIF graphic downloads one line of pixels at a time from top to bottom, and browsers display each line of the image as it gradually builds on the screen.

In interlaced GIF files the image data is stored in a format that allows browsers to build a low-resolution version of the full-sized GIF picture on the screen while the file is downloading. The most important benefit of interlacing is that it gives the reader a preview of the full area of the picture while the picture downloads into the browser.

Interlacing is best for larger GIF images such as illustrations and photographs. Interlacing is a poor choice for small GIF graphics such as navigation bars, buttons, and icons.

Half of figure downloaded,
non-interlaced GIF

Half of figure downloaded,
interlaced GIF



Animated GIF

For combining multiple GIF images into a single file to create animation, GIF file format is used.

There are a number of drawbacks to this functionality.

The GIF format applies no compression between frames, so if you are combining four 30-kilobyte images into a single animation, you will end up with a 120 KB GIF file to push through the wire.

Another drawback of GIF animations is that there are no interface controls for this file format, GIF animations play whether you want them to not. And if looping is enabled, the animations play again and again and again.

JPEG Graphics

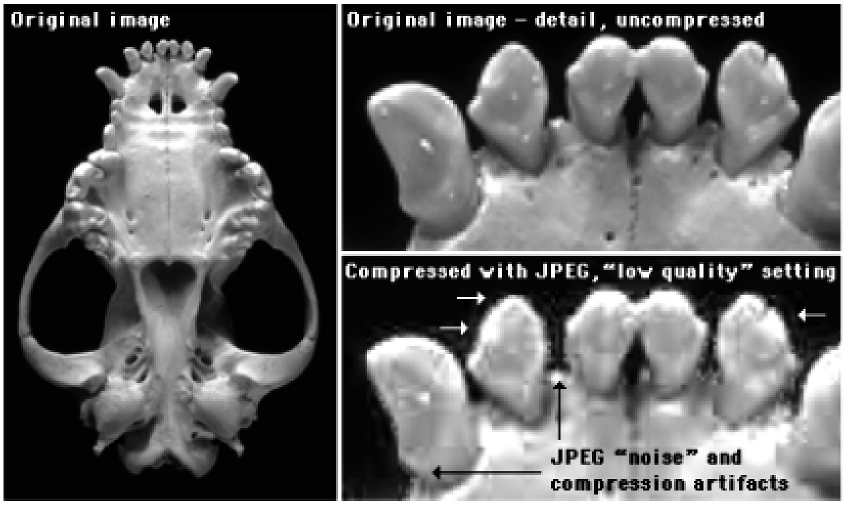
The other graphic file format commonly used on the Web to minimize graphics file sizes is the Joint Photographic Experts Group (JPEG) compression scheme. Unlike, GIF graphics, JPEG images are full-colour images (24 bit, or "true color"). JPEG images find great acceptability among photographers, artists, graphic designers, medical imaging specialists, art historians, and other groups for whom image quality is paramount and where colour fidelity cannot be compromised.

JPEG compression uses a complex mathematical technique called a discrete cosine transformation to produce a sliding scale of graphics compression. The degree of compression can be chosen but it is inversely proportional to image. The more you squeeze a picture with JPEG compression, the more you degrade its quality.

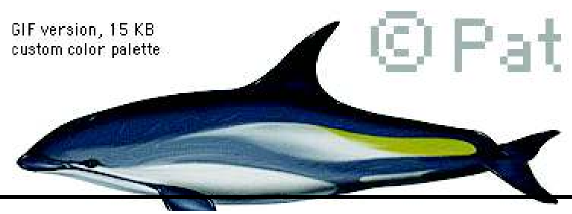
JPEG can achieve incredible compression ratios up to 1:100.

This is possible because the JPEG algorithm discards "unnecessary" data as it compresses the image, and it is thus called a "lossy" compression technique.

Notice in the example below, how increasing the JPEG compression progressively degrades the details of the image:



Another example of JPEG compression is shown below. Note, the extensive compression noise and distortion present in the bottom dolphin — the download time saved is not worth the degrading of the images.



2.4.4 Uses for GIF and JPEG Files

Netscape Navigator, Microsoft Internet Explorer, and most other browsers support both GIF and JPEG graphics.

In theory, you could use either graphic format for the visual elements of your Web pages. In practice, however, most Web developers still favour the GIF format for most page design elements, diagrams, and images that must not dither on 8-bit display screens.

Designers choose the JPEG format mostly for photographs and complex "photographic" illustrations.

Advantages of GIF Files

- GIF is the most widely supported graphics format on the Web.
- GIFs of diagrammatic images look better than JPEGs.
- GIF supports transparency and interlacing.

Advantages of JPEG Images

- Huge compression ratios mean faster download speeds.
- JPEG produces excellent results for most photographs and complex images.
- JPEG supports full-colour (24-bit, "true color") images.

Other File Formats

BMP/DIB/RLE File Formats

These are known as device independent bitmap files. They exist in two different formats a) OS2 format and b) Windows format. BMP is the standard MS-windows raster format created with windows paintbrush and used as wallpaper for the background while running windows. DIB or device independent bitmap file are mainly applied in computer multimedia systems and can be used as image files in the windows environment. RLE or run length coding files are actually DIB files that use one of the RLE compression routines.

IMG/MAC/MSP File Formats

IMG files were originally designed to work with GEM paint program and can handle monochrome and grey level images only.

MAC files are used in Macintosh Mac Paint application. MAC file format has two basic options:

- Ported Mac Paint files that include a Mac Binary header, and

- are used with PFS first publisher with no header.

MSP files originated in the pre-historic MS-Paint and can be converted into BMP files.

WPG

WPG or word perfect graphic file is used by Word Perfect. It first appeared with the release of word perfect 5.0. These files can contain bitmaps, line arts and vector graphics. WPG specifications allows files up to 256 colours.

IFF

Amiga Interchange File Format is used to transfer documents to and from Commodore Amiga Computers. The IFF format is extremely flexible and allows images and text to be stored inside an IFF file. The format can also be created on a PC but the extension of file name will change to **LBM or CE.**

PIXEL PAINT

The pixel paint file format allows a document to be opened in the pixel paint and pixel paint professional graphics application. This format allows you to specify the image size or canvas. It also enable you to decide whether you want the image to appear in the center or the upper left corner of the canvas when the document is opened.

JAS

The JAS file formats were designed to create the smallest possible image files for 24bits per pixel image and 8 bit per pixel gray scaled images. It uses a discrete cosine transformation to alter and compress the image data. This type of storage and retrieval results in some loss of image data and this loss is dependant on the compression level selected by the application.

TIFF

Tagged Image file format is used mainly for exchanging documents between different applications and different computers.

It was primarily designed to become the standard file format. The result of this design provided the flexibility of an infinite numbers of possibilities of how a TIFF image can be saved.

This format uses 6 different encoding routines:

- No compression
- Huffman
- Pack Bits
- LZW
- Fax Group 3
- Fax Group 4

In addition, it differentiates between three types of images in three different categories:

- Black and White
- Grey Scaled
- Colored

☞Check Your Progress 3

1) What is computer graphics ?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

2) What are the various types of graphic images?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

3) Why file compression techniques is beneficial in computer graphics ?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

4) JPEG is ideal for faster downloads. Justify.
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………
………

2.5 AUDIO AND VIDEO

Audio and Video are working as ear and eye of multimedia. Both of them are heavily contributing to any multimedia application. Let us discuss something about the association of these fields with multimeida.

2.5.1 Sound and Audio

Sound is a mechanical energy disturbance that propagates through matter as a wave. Sound is characterised by the various properties which are frequency, wavelength, period, amplitude and velocity or speed.

Noise and sound often mean the same thing but a noise is an unwanted sound. In science and engineering, noise is an undesirable component that obscures a signal.

Humans perceive sound by the sense of hearing. By sound, we commonly mean the vibrations that travel through air and can be heard by humans. However, scientists and engineers use a wider definition of sound that includes low and high frequency vibrations in air that cannot be heard by humans, and vibrations that travel through all forms of matter, gases, liquids and solids. The matter that supports sound is called the medium.

Sound propagates as waves of alternating pressure, causing local regions of compression and rarefaction. Particles in the medium are displaced by the wave and oscillate as result of the displacement. The scientific study of sound is called

acoustics. The sound portion of a program, or, a track recorded on a videotape which contains sound, music, or narration is called **Audio**.

2.5.2 Analog Sound vs. Digital Sound

Sound engineers have been debating the respective merits of analog and digital sound reproduction ever since the appearance of digital sound recordings. This is one of the never ending controversies in the field, much like that comparison of vacuum tube amplifiers against those of solid state (transistor) electronics. In consumer audio, the opposition is usually between vinyl LP recordings and compact discs.

An analog recording is one where the original sound signal is modulated onto another physical signal carried on some media or the groove of a gramophone disc or the magnetic field of a magnetic tape. A physical quantity in the medium (e.g., the intensity of the magnetic field) is directly related to the physical properties of the sound (e.g., the amplitude, phase and possibly direction of the sound wave.)

A digital recording, on the other hand is produced by first encoding the physical properties of the original sound as digital information which can then be decoded for reproduction. While it is subject to noise and imperfections in capturing the original sound, as long as the individual bits can be recovered, the nature of the physical medium is of minimum consequence in recovery of the encoded information.

A damaged digital medium, such as a scratched compact disc may also yield degraded reproduction of the original sound, due to the loss of some digital information in the damaged area (but not due directly to the physical damage of the disc).

Arguments made in favour of Analog Sound

- Shape of the waveforms: sound reconstructed from digital signals is claimed to be harsher and unnatural compared to analog signals.
- Lower distortion for low signal levels.
- Absence of quantisation noise.
- Absence of aliasing.
- Not subject to jitter.
- Euphonic characteristics.

Arguments made in favor of Digital Sound

- Lower noise floor.
- Dynamic range.
- Signal to noise ratio.
- Absence of generation loss.
- Resistance to media deterioration.
- Immunity to wow and flutter.
- Ability to apply redundancy like error-correcting codes, to prevent data loss.

Digital audio comprises audio signals stored in a digital format. Specifically, the term encompasses the following:

1) Audio conversion:
    1. Analogue to digital conversion (ADC)
    2. Digital to analogue conversion (DAC).

An **analog-to-digital converter** (abbreviated **ADC**, **A/D** or **A to D**) is an electronic circuit that converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital-to-analog converter (DAC).

Typically, an ADC is an electronic device that converts an input analog voltage to a digital number. The digital output may be using different coding schemes, such as binary and two's complement binary. However, some non-electronic or only partially electronic devices, such as shaft encoders, can also be considered to be ADCs.

2) Audio signal processing - processing the digital signal to perform sample rate conversion.

Audio signal processing, sometimes referred to as **audio processing**, is the processing of auditory signals, or sound represented in digital or analog fromat. An analog representation is usually electrical; a voltage level represents the air pressure waveform of the sound. Similarly, a digital representation is in the form of pressure wave-form represented as a sequence of binary numbers, which permits digital signal processing.

The focus in audio signal processing is most typically a mathematical analysis of the parts of the signal that are audible. For example, a signal can be modified for different purposes such that the modification is controlled by the auditory domain.

Processing methods and application areas include storage, level compression, data compression, transmission, enhancement (e.g., equalisation, filtering, noise cancellation, echo or reverb removal or addition, etc.), source separation, sound effects and computer music.

3) Storage, retrieval, and transmission of digital information in an audio format such as CD, MP3, Ogg Vorbis, etc.

An **audio format** is a medium for storing sound and music. The term is applied to both the physical medium and the format of the content.

Music is recorded and distributed using a variety of audio formats, some of which store additional information.

Sound inherently begins and ends as an analogue signal, and in order for the benefits of digital audio to be realised, the integrity of the signal during transmission must be maintained. The conversion process at both ends of the chain must also be of low loss in order to ensure sonic fidelity.

In an audio context, the digital ideal would be to reproduce signals sounding as near as possible to the original analogue signal. However, conversion is “lossy”: conversion and compression algorithms deliberately discard the original sound information, mainly harmonics, outside the theoretical audio bandwidth

Digital information is also lost in transfer through misreading, but can be “restored” by error correction and interpolation circuitry. The restoration of the original music waveforms by decompression during playback should be exactly the same as the compression process. However, a few harmonics such as the upper harmonics which have been discarded can never be restored, with complete accuracy or otherwise. Upon its re-conversion into analogue via the amplifier/loudspeaker, the scheme relies heavily on the human brain to supply the missing sound during playback.

Pulse-code Modulation (PCM) is by far the most common way of representing a digital signal. It is simple and is compressed. A PCM representation of an analogue signal is generated by measuring (sampling) the instantaneous amplitude of the analogue signal, and then quantising the result to the nearest bit. However, such rounding contributes to the loss of the original information.

Digital audio technologies

- Digital Audio Tape (DAT)
- DAB (Digital Audio Broadcasting)
- Compact disc (CD)
- DVD DVD-A
- Minidisc (obsolete as of 2005)
- Super audio compact disc
- Digital audio workstation
- Digital audio player
- and various audio file formats

2.5.3 Audio File Formats

An **audio file format** is a container format for storing audio data on a computer system. There are numerous file formats for storing audio files.

The general approach towards storing digital audio formats is to sample the audio voltage in regular intervals (e.g. 44,100 times per second for CD audio or 48,000 or 96,000 times per second for DVD video) and store the value with a certain resolution (e.g. 16 bits per sample in CD audio). Therefore sample rate, resolution and number of channels (e.g. 2 for stereo) are key parameters in audio file formats.

Types of Formats

It is important to distinguish between a file format and a codec. Though most audio file formats support only one audio codec, a file format may support multiple codecs, as AVI does.

There are three major groups of audio file formats:

- common formats, such as WAV, AIFF and AU.
- formats with lossless compression, such as FLAC, Monkey's Audio (filename extension APE), WavPack, Shorten, TTA, Apple Lossless and lossless Windows Media Audio (WMA).
- formats with lossy compression, such as MP3, Vorbis, lossy Windows Media Audio (WMA) and AAC.

Uncompressed / Common Audio Format

There is one major uncompressed audio format: PCM. It is usually stored as a .wav on Windows. WAV is a flexible file format designed to store more or less any combination of sampling rates or bitrates. This makes it an adequate file format for storing and archiving an original recording. A lossless compressed format would require more processing for the same time recorded, but would be more efficient in terms of space used. WAV, like any other uncompressed format, encodes all sounds, whether they are complex sounds or absolute silence, with the same number of bits per unit of time.

The WAV format is based on the RIFF file format, which is similar to the IFF format.

Lossless Audio Formats

Lossless audio formats (such as TTA and FLAC) provide a compression ratio of about 2:1, sometimes more. In exchange, for their lower compression ratio, these codecs do not destroy any of the original data. This means that when the audio data is uncompressed for playing, the sound produced will be identical to that of the original

sample. Taking the free TTA lossless audio codec as an example, one can store up to 20 audio CDs on one single DVD-R, without any loss of quality. The negative aspect of this was that this DVD would not only require a DVD reader but a system which could decode the chosen codec as well for playing. This will most likely be a home computer. Although these codecs are available for free, one important aspect of choosing a lossless audio codec is hardware support. It is in the area of hardware support that FLAC is ahead of the competition. FLAC is supported by a wide variety of portable audio playback devices.

Lossy Audio Formats

Lossy file formats are based on sound models that remove audio data that humans cannot or can hardly hear, e.g. a low volume sound after a big volume sound. MP3 and Vorbis are popular examples. One of the most popular lossy audio file formats is MP3, which uses the MPEG-1 audio layer 3 codec to provide acceptable lossy compression for music files. The compression is about 10:1 as compared to uncompressed WAV files (in a standard compression scheme), therefore, a CD with MP3 files can store about 11 hours of music, compared to 74 minutes of the standard CDDA, which uses uncompressed PCM.

There are many newer lossy audio formats and codecs claiming to achieve improved compression and quality over MP3. Vorbis is an unpatented, free codec.

Multiple Channels

Since the 1990s, movie theatres have upgraded their sound systems to surround sound systems that carry more than two channels. The most popular examples are Advanced Audio Coding or AAC (used by Apple's iTunes) and Dolby Digital, also known as AC-3. Both codecs are copyrighted and encoders/decoders cannot be offered without paying a licence fee. Less common are Vorbis and the recent MP3-Surround codec. The most popular multi-channel format is called 5.1, with 5 normal channels (front left, front middle, front right, back left, back right) and a subwoofer channel to carry low frequencies only.

2.5.4 Image Capture Formats

Video cameras come in two different image capture formats: interlaced and progressive scan.

Interlaced Scan

Interlace is a technique of improving the picture quality of a video transmission without consuming any extra bandwidth. It was invented by the RCA engineer **Randall Ballard** in the late 1920s .

It was universally used in television until the 1970s, when the needs of computer monitors resulted in the reintroduction of progressive scan. While interlace can improve the resolution of still images, on the downside, it causes flicker and various kinds of distortion. Interlace is still used for all standard definition TVs, and the 1080i HDTV broadcast standard, but not for LCD, micromirror (DLP, or plasma displays).

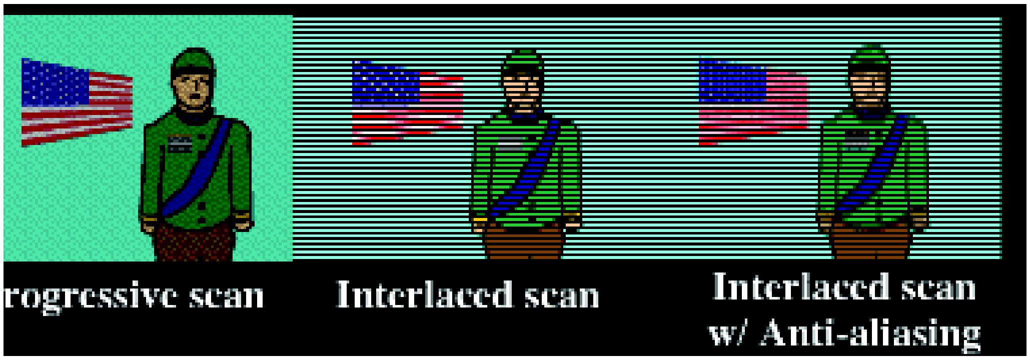
These devices require some form of deinterlacing which can add to the cost of the set.

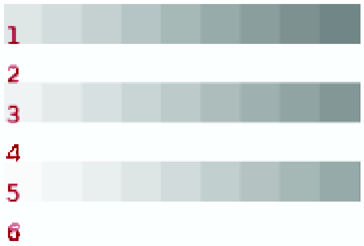
With progressive scan, an image is captured, transmitted and displayed in a path similar to the text on a page: line by line, from top to bottom.

The interlaced scan pattern in a CRT (cathode ray tube) display would complete such a scan too, but only for every second line and then the next set of video scan lines would be drawn within the gaps between the lines of the previous scan.

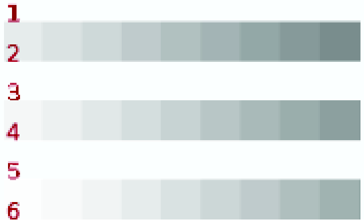
Such scan of every second line is called a field.

The afterglow of the phosphor of CRT tubes, in combination with the persistence of vision results in two fields being perceived as a continuous image which allows the viewing of full horizontal detail but with half the bandwidth which would be required for a full progressive scan while maintaining the necessary CRT refresh rate to prevent flicker.





Odd field



Even Feild

Since, after glow or persistence of vision plays an important part in iterlaced scan, only CRTs can display interlaced video directly – other display technologies require some form of deinterlacing.

In the 1970s, computers and home video game systems began using TV sets as display devices. At this point, a 480-line NTSC signal was well beyond the graphics abilities of low cost computers, so these systems used a simplified video signal in which each video field scanned directly on top of the previous one, rather than each line between two lines of the previous field.

By the 1980s computers had outgrown these video systems and needed better displays. Solutions from various companies varied widely. Because PC monitor signals did not need to be broadcast, they could consume far more than the 6, 7 and 8 MHz of bandwidth that NTSC and PAL signals were confined to.

In the early 1990s, monitor and graphics card manufacturers introduced newer high resolution standards that once again included interlace. These monitors ran at very high refresh rates, intending that this would alleviate flicker problems. Such monitors proved very unpopular. While flicker was not obvious on them at first, eyestrain and lack of focus nevertheless became a serious problem. The industry quickly abandoned this practice, and for the rest of the decade all monitors included the assurance that their stated resolutions were "non-interlace".

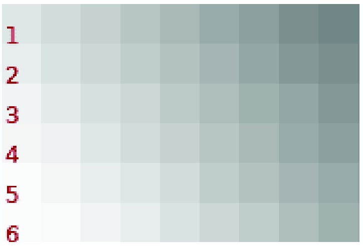
Application

Interlacing is used by all the analogue TV broadcast systems in current use:

- **PAL**: 50 fields per second, 625 lines, odd field drawn first
- **SECAM**: 50 fields per second, 625 lines
- **NTSC**: 59.94 fields per second, 525 lines, even field drawn first

Progressive Scan

Progressive or **non-interlaced scanning** is a method that displays, stores, or transmits moving images in which, the lines of each frame are drawn in sequence. This is in contrast to the interlacing used in traditional television systems.



Progressive Scan

Progressive scan is used for most CRT computer monitors. (Other CRT-type displays, such as televisions, typically use interlacing.) It is also becoming increasingly common in high-end television equipment.

Advantages of progressive scan include:

- Increased vertical resolution. The perceived vertical resolution of an interlaced image is usually equivalent to multiplying the active lines by about 1.6
- No flickering of narrow horizontal patterns
- Simpler video processing equipment
- Easier compression

2.5.5 Digital Video

Digital video is a type of video recording system that works by using a digital, rather than analog, representation of the video signal. This generic term is not to be confused with the name DV, which is a specific type of digital video. Digital video is most often recorded on tape, then distributed on optical discs, usually DVDs. There are exceptions, such as camcorders that record directly to DVDs Digital8 camcorders which encode digital video on conventional analog tapes, and the most recent JVC Everio G camcorders which record digital video on hard disks.

Digital video is not like normal analogue video used by everyday televisions. To understand how digital video works it is best to think of it as a sequence of non-interlaced images, each of which is a two-dimensional frame of picture elements or pixels. Present day analogue television systems such as:

- The National Television Standards Committee (NTSC), used in North America and Japan
- Phase Alternate Line (PAL), used in western Europe,

employ line interlacing. Systems that use line interlacing alternately scan odd and even lines of the video, which can produce images when analogue video is digitized.

In case of Digital video, there are two terms associated with each pixel, luminance and chrominance. The luminance is a value proportional to the pixel's intensity. The chrominance is a value that represents the colour of the pixel and there are a number of representations to choose from. Any colour can be synthesised by an appropriate mixture of three properly chosen primary colours. Red, Green and Blue (RGB) are usually chosen for the primary colours.

When an analogue signal is digitised, it is quantised. Quantisation is the process by which a continuous range of values from an input signal is divided into non-overlapping discrete ranges and each range assigned a unique symbol. A digitised monochrome photograph might, for example, contain only 256 different kinds of pixel. Such an image would be said to have a pixel depth of 8 bits. A higher quality image might be quantised allowing 24 bits per pixel.

Digital video can be characterised by a few variables:

Frame rate: The number of frames displayed per second. The illusion of motion can be experienced at frame rates as low as 12 frames per second, but modern cinema uses 24 frames per second, and PAL television 25 frames per second.

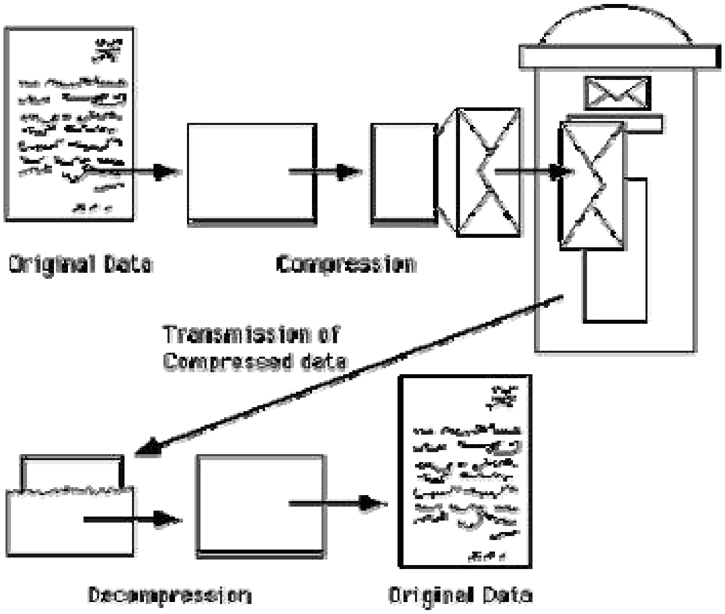
Frame dimensions: The width and height of the image expressed in the number of pixels. Digital video comparable to television requires dimensions of around 640 x 480 pixels.

Pixel depth: The number of bits per pixel. In some cases it might be possible to separate the bits dedicated to luminance from those used for chrominance. In others, all the bits might be used to reference one of a range of colours from a known palette.

The table below illustrates possible values of these parameters for typical applications of digital video.

Application	Frame rate	Dimensions	Pixel Depth
Multimedia	15	320 x 240	16
Entertainment TV	25	640 x 480	16
Surveillance	5	640 x 480	12
Video Telephony	10	320 x 240	12
HDTV	25	1920 x 1080	24

Advances in compression technology more than anything else have led to the arrival of the video to the desktop and hundreds of channels to homes.



Compression is a reversible conversion of data to a format that requires fewer bits, usually performed so that the data can be stored or transmitted more efficiently. The size of the data in compressed form (C) relative to the original size (O) is known as the compression ratio (R=C/O). If the inverse of the process, decompression, produces an exact replica of the original data then the compression is lossless.

Lossy compression, usually applied to image data, does not allow reproduction of an exact replica of the original image, but has a higher compression ratio. Thus, lossy compression allows only an approximation of the original to be generated.

Compression is analogous to folding a letter before placing it in a small envelope so that it can be transported more easily and cheaply (as shown in the figure). Compressed data, like the folded letter, is not easily read and must first be decompressed, or unfolded, to restore it to its original form.

The success of data compression depends largely on the data itself and some data types are inherently more compressible than others. Generally some elements within the data are more common than others and most compression algorithms exploit this property, known as redundancy. The greater the redundancy within the data, the more successful the compression of the data is likely to be. Fortunately, digital video contains a great deal of redundancy and thus, is very suitable for compression.

A device (software or hardware) that compresses data is often know as an encoder or coder, whereas a device that decompresses data is known as a decoder. A device that acts as both a coder and decoder is known as a codec.

Compression techniques used for digital video can be categorised into three main groups:

- General purpose compression techniques can be used for any kind of data.
- Intra-frame compression techniques work on images. Intra-frame compression is compression applied to still images, such as photographs and diagrams, and exploits the redundancy within the image, known as spatial redundancy. Intra-frame compression techniques can be applied to individual frames of a video sequence.

Inter-frame compression techniques work on image sequences rather than individual images. In general, relatively little changes from one video frame to the next. Inter-frame compression exploits the similarities between successive frames, known as temporal redundancy, to reduce the volume of data required to describe the sequence.

2.5.6 Need for Video Compression

The high bit rates that result from the various types of digital video make their transmission through their intended channels very difficult. Even entertainment video with modest frame rates and dimensions would require bandwidth and storage space far in excess of that available on the CD-ROM. Thus, delivering consumer quality video on compact disc would be impossible.

Similarly, the data transfer rate required by a video telephony system is far greater than the bandwidth available over the plain old telephone system (POTS). Even if high bandwidth technology (e.g. fiber-optic cable) was in place, the per-byte-cost of transmission would have to be very low before it is feasible to use for the staggering amounts of data required by HDTV.

Lastly, even if the storage and transportation problems of digital video were overcome, the processing power needed to manage such volumes of data would make the receiver hardware highly optimized.

Although significant gains in storage, transmission, and processor technology have been achieved in recent years, it is primarily the reduction of the amount of data that needs to be stored, transmitted, and processed that has made widespread use of digital video a possibility.

This reduction of bandwidth has been made possible by advances in compression technology.

2.5.7 Video File Formats

DV Encoder Types

When DV is captured into a computer it is stored in an AVI file, which is Microsoft's standard file format for video files. Video support in Windows is provided by DirectShow, a high performance 32 bit interface.

Digital video can be stored in two formats, DV Encoder Type 1 and DV Encoder Type 2.

DV Encoder Type 1

The standard DV bit stream interfaces the video and audio streams together. This format is fully supported by DirectShow which accepts this interleaved stream and provides splitter and multiplexer filters to isolate or interlace the video and audio streams from DV. With an Encoder Type 1 AVI file the raw DV interleaved data stream is simply written into the file.

DV Encoder Type 2

Encoder Type 2 produces a VfW compatible AVI file format. This file has separate streams for video and audio and it can also be processed by DirectShow. The advantage of creating an Encoder Type 2 file is that the file can be read by the older applications that do not support DirectShow.

Other Video File Formats

There are numerous other formats for storing video in digital formats. These formats are generally used for the storage and viewing of video by and on computer systems (with the exception of the MPEG formats).

AVI CODEC Formats

There are numerous AVI file formats other than the DV Types 1 and 2 formats discussed earlier. All these other formats involve the use of Compressor / DE-Compressors (CODECs) to read and write the AVI file. All invariably compress the video by reducing frame size from the standard 720 x 480 to 240 x 160 or smaller, by reducing the number of frames per second and by washing out colour, contrast and intensity. The resulting file size may be attractive, but the quality is usually quite poor. CinePac and Indeo are commonly used CODECs.

MPEG-1

MPEG-1 (Moving Picture Experts Group format 1) is an industry standard encoding format that is widely used. It's normal format is a frame size of 352 x 240 and a constant bit stream of around one megabit per second, a rate well within that of any CD player. MPEG-1 at this size consumes around 10 megabytes for each minute of video, so a typical CD can hold about 1 hour of video.

MPEG-1 is roughly equivalent to VHS in quality, although one might not think so, when one watches the video on a computer. Video CDs (VCDs) use the MPEG-1 format, and look good when viewed on a television.

MPEG-2

MPEG-2 is the standard used by DVD and is of a much higher quality than MPEG-1. This format provides for 720 x 480 resolution and with much less loss of detail over MPEG-1. However, the file sizes are 3 to 4 times larger than MPEG-1.

A DVD can contain many hours of MPEG-2 video, but the cost of the DVD writer is still high. MPEG-2 on a CD is possible, using a format called that SVCD but that can only contain about 20 minutes of video.

Quicktime

Quicktime is the video format devised by and used by Apple and can be used at varying quality and file sizes. It is quite widely used and has influenced the design of the MPEG formats.

Real Video

Real video is a streaming video format used for distributing video in real-time over the internet. With streaming video, you do not have to download the complete file before beginning to watch it. Rather the viewer will download the first section of the video while the remainder downloads in the background.

☞Check Your Progress 4

1) Compare analog and digital sounds.

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   ……………………………………………………………………………………

   …………

2) What are various types of audio file formats?

……………………………………………………………………………………

……………..…………………………………………………………………

……………..……………………………………………………………………

……………………………………………………………………………………

…………

3) What are the various types of video file formats?

……………………………………………………………………………………

…………..……………………………………………………………………

……………..……………………………………………………………………

……………………………………………………………………………………

…………

4) Why is compression required in digital video ?

……………………………………………………………………………………

…………..……………………………………………………………………

……………..……………………………………………………………………

……………………………………………………………………………………

…………

5) Explain interlaced and progressive scan in image capturing techniques.

……………………………………………………………………………………

…………..……………………………………………………………………

……………..……………………………………………………………………

……………………………………………………………………………………

…………

2.6 MULTIMEDIA TOOLS

In this section, we will emphasise on various tools used in the field of multimedia.

2.6.1 Basic Tools

The basic toolset for building multimedia projects, contains, one or more authoring systems and various applications for text, images, sounds and motion video editing.

A few additional applications are useful for capturing images from the screen, changing file formats and moving files among computers when you are part of a team. These are basically tools for the housekeeping tasks that make your creativity and productivity better.

The software in your multimedia toolkit and your skill at using it will determine the kind of multimedia work you can do and how fine and fancy you can render it.

2.6.2 Types of Basic Tools

Various types of basic tools for creating and editing multimedia elements are :

- Painting and Drawing tools
- Image editing tools
- OCR software
- 3-D Modeling and Animation tools
- Sound editing programs
- Animation, Video and Digital movies

Painting and Drawing Tools

Painting software is dedicated to producing crafted bitmapped images. Drawing software like Corel Draw and Canvas is dedicated to producing vector based line art easily printed to paper using Postscript or another page mark up system such as Quick Draw.

Main features / criteria for selection are:

- Intuitive graphical interface with pull down menus, status bars, palette control and dialog boxes for quick logical selection.
- Scalable dimensions for resizing, stretching and distorting.
- Paint tools to create geometric shapes.
- Ability to pour a colour, pattern or gradient.
- Ability to paint with patterns and clip arts.
- Customisable pen and brush shape and sizes.
- Eyedropper tools for colour sampling.
- Auto trace tool for converting bitmapped images into vector based outlines.
- Multiple undo capabilities.
- Support for scalable text fonts.
- Painting features with anti-aliasing, air brushing, color washing, blending, masking etc.
- Support for third party special effects.
- Object and layering capabilities.
- Zooming for magnified pixel editing.
- All common colour depths.
- Good file importing and exporting capabilities.

Image Editing Tools

These are specialise and powerful tools for enhancing and re-touching existing bitmapped images. These applications also provide many of the features and tools of the painting and drawing programs and can be used for creating images from scratch as well as images digitised from scanners, video frame grabbers, digital camera, clip art files or original art work files created with a drawing package.

Features Typical of image editing applications are:

- Conversion of major image data types and industry standard file formats.
- Direct input from scanners etc.
- Employment of virtual memory scheme.
- Multiple window scheme.
- Image and balance control for brightness, contrast etc.
- Masking undo and restore features.
- Multiple video, Anti-aliasing, sharpening and smoothing controls.
- Colour mapping controls.
- Geometric transformations.
- All colour palettes.
- Support for third party special effects plugins.
- Ability to design in layers that can be combined, hidden and recorded.

Optical Character Recognition Software (OCR)

Often, you will have printed matter and other text to incorporate into your project but no electronic text file. With OCR software, a flat bed scanner and your computer you can save many hours of rekeying printed words and get the job done faster and more accurately than a roomful of typists.

OCR software turns bitmapped characters into electronically recognizable ASCII text. A scanner is typically used to create the bitmap. Then, the software breaks the bitmap into chunks according to whether it contains text or graphics by examining the texture and density of areas of the bitmap and by detecting edges. The text areas of the bitmap are then converted to ASCII characters using probability and expert system algorithm. Most OCR application claim 99 percent accuracy when reading 8 to 36 point characters at 300 dpi and can reach processing speeds of about 150 character per second.

3-D Modeling and Animation Tools

With 3-D modeling software, objects rendered in perspective appear more realistic. One can create stunning scenes and wander through them, choosing just the right lighting and perspective for your final rendered image. Powerful modeling packages such as Macromedia's Extreme 3 D, Autodesk's 3 D Studio Max. Strata Vision's 3D, Specular's Logo motion and Infini-D and Caligari's truespace are also bundled with assortments of pre-rendered 3-D clip art objects such as people, furniture, buildings, cars, aero plane, trees and plants.

Features for good 3-D modeling software are :

- Multiple window that allow you to view your model in each dimension.
- Ability to drag and drop primitive shapes into a scene.
- Create and sculpt organic objects from scratch.
- Lathe and extrude features.
- Colour and texture mapping.
- Ability to add realistic effects such as transparency, shadowing and fog.
- Ability to add spot, local and global lights, to place them anywhere and manipulate them for special effects.
- Unlimited cameras with focal length control.

Sound Editing Programs

Sound editing tools for both digitised and MIDI sound lets you see music as well as hear it. By drawing a representation of a sound in fine increments, whether a score or a waveform, you can cut, copy, paste and otherwise edit segments of it with great precision – something impossible to do in real time with music playing.

System sounds are beeps used to indicate an error, warning or special user activity. Using sound editing software, you can make your own sound effects and install them as system beeps.

Animation, Video and Digital Movies

Animations and digital video movies are sequences of bitmapped graphic scenes or frames, rapidly played back. But animations can also be made within the authoring system by rapidly changing the location of the object to generate an appearance of motion. Most authoring tools adapt either a frame or object oriented approach to animation but rarely both.

Movie making tools take advantage of QuickTime and Microsoft Video for Windows also known as AVI or Audio Video Interleaved technology and let you create, edit

and present digitised video motion segments usually in a small window in your project.

To make movies from video you need special hardware to convert the analog video signal to digital data. Movie making tools such as Premiere, Video Shop and Media Studio Pro let you edit and assemble video clips captured from camera, tape and other digitised movie segments, animations, scanned images and from digitised audio and MIDI files. The completed clip usually with added transition and visual effects can then be played back either stand alone or windowed within your project.

Morphing is an animation technique that allows you to dynamically blend two still images creating a sequence of in-between pictures that when played back rapidly in Quick Time, metamorphoses the first image into the second. For example a racing car transforms into a tiger, and a daughter's face becomes her mother's.

Accessories

A Screen Grabber is an essential accessory. Bitmap images are so common in multimedia, that it is important to have a tool for grabbing all or part of the screen display so that you can import it into your authoring system or copy it into an image editing application. Screen grabbing to the clipboard lets you move a bitmapped image from one application to another without the cumbersome steps of exporting the image to a file and then importing it back to the destination.

Another useful accessory is Format Converter which is also indispensable for projects in which your source material may originate on Macintoshes, PCs, Unix Workstations or even mainframes.

2.6.3 Authoring Tools

Authoring tools usually refers to computer software that helps multimedia developers create products. Authoring tools are different from computer programming languages in that they are supposed to reduce the amount of programming expertise required in order to be productive. Some authoring tools use visual symbols and icons in flowcharts to make programming easier. Others use a slide show environment.

Authoring tools help in the preparation of texts. Generally, they are facilities provided in association with word processing, desktop publishing, and document management systems to aid the author of documents. They typically include, an on-line dictionary and thesaurus, spell-checking, grammar-checking, style-checking, and facilities for structuring, integrating and linking documents.

Also known as Authorware, it is a program that helps you write hypertext or multimedia applications. Authoring tools usually enable you to create a final application merely by linking together objects, such as a paragraph of a text, an illustration, or a song. By defining the objects' relationships to each other, and by sequencing them in an appropriate order, authors (those who use authoring tools) can produce attractive and useful graphics applications.

The distinction between authoring tools and programming tools is not clear-cut. Typically, though, authoring tools require less technical knowledge to master and are used exclusively for applications that present a mixture of textual, graphical, and audio data.

Multi media authoring tools provide the important framework you need for organising and editing the elements of your multi media project including graphics, sounds, animations and video clips. Authoring tools are used for designing interactivity and user interface, for presenting your project on screen and for assembling multimedia elements into a single cohesive project.

Authoring software provides an integrated environment for binding together the contents of your project. Authoring systems typically include the ability to create, edit and import specific types of data, assemble raw data into a playback sequence or a cue sheet and provide a structured method or language for responding to user input.

2.6.4 Types of Authoring Tools

Authoring tools are grouped based on metaphor used for sequencing or organising multimedia elements and events:

- Card or Page Based Tools
- Icon Based or Event Driven Tools
- Time Based and Presentation Tools
- Object Oriented Tools

i) **Card or Page Based Tools**

In these authoring systems, elements are organised as pages of a book or a stack of cards. Thousands of pages or cards may be available in the book or stack. These tools are best used when the bulk of your content consists of elements that can be viewed individually, like the pages of a book or cards in a card file.

The authoring system lets you link these pages or cards into organised sequences. You can jump on command to any page you wish in the structured navigation pattern. Card or Page based authoring systems allow you to play the sound elements and launch animation and digital video.

ii) **Icon Based or Event Driven Tools**

In these authoring systems, multimedia elements and interaction cues or events are organised as objects in a structural framework or process. Icon – based, event driven tools simplify the organisation of your project and typically displays flow diagrams of activities along branching paths. In complicated navigational structures, this charting is particularly useful during development.

iii) **Time Based and Presentation Tools**

In these authoring systems, elements and events are organised along a timeline, with resolutions as high as 1/30 second. Time based tools are best used when you have a message with a beginning and an end. Sequentially organised graphic frames are played back at speed that, you can set. Other elements such as audio events are triggered at a given time or location in the sequence of events. The more powerful time based tools lets your program jump to any location in a sequence thereby adding navigation and interactive control.

iv) **Object Oriented Tools**

In these authoring systems, multimedia elements and events become objects that live in hierarchical order of parent and child relationship. Messages are passed among these objects, ordering them to do things according to the properties or modifiers assigned to them. In this way, for example, Jack may be programmed to wash dishes every Friday evening and does so when he gets the message from his wife. Objects typically take care of themselves. Send them a message and they do their thing without external procedures and programming. Object – oriented tools are particularly useful for games, which contain many components with many "personality".

2.6.5 Multimedia Tool Features

Common to nearly all multimedia tool platforms are a number of features for encapsulating the content, presenting the data, obtaining user input and controlling the execution of the product. These feature include:

- Page
- Controls
    - Navigation
    - Input
    - Media Controls
- Data
    - Text
    - Graphics
    - Audio
    - Video
    - Live Audio/Video
    - Database
- Execution
    - Linear Sequenced
    - Program controlled
    - Temporal Controlled
    - Inter activity Controlled

☞Check Your Progress 5

1) What are the basic tools of multimedia?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………

2) What is the selection criteria for image editing tools ?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
………

3) What are multimedia authoring tools ?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
………

4) What are the types or categories of authoring tools ?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
………

2.7 SUMMARY

Multimedia as the name suggests MULTI and MEDIA uses several media (e.g. text, audio, graphics, animation, video) to convey information. Multimedia also refers to the use of computer technology to create, store, and experience multimedia content.

In this unit, we have tried to understand the concept of multimedia, its applications in various fields like education, training, business, entertainment to name a few.

Another section deals with defining objects for multimedia systems in order to understand the nuts and bolts of multimedia technology like still pitures, graphics, animation, sound, video etc.

These objects of multimedia system need to be transmitted, hence, there is a need for their compression and various techniques of compression for optimum bandwidht usage.

The last section deals with the basic toolset for building multimedia projects which contains one or more authoring systems and various applications for texts, images, sounds and motion video editing. It also addresses the questions – What is the basic hardware and software needed to develop and run multimedia technology and applications?

The software in your multimedia toolkit and your skill at using it determines what kind of multimedia work you can do and how fine and fancy you can render it.

2.8 SOLUTIONS / ANSWERS

Check Your Progress 1

1) **Hypertext** - Hypertext is conceptually the same as a regular text - it can be stored, read, searched, or edited - with an important difference: hypertext is text with pointers to other text. The browsers let you deal with the pointers in a transparent way -- select the pointer, and you are presented with the text that is pointed to.

   Hypermedia - Hypermedia is a superset of hypertext. Hypermedia documents contain links not only to other pieces of text, but also to other forms of media - sounds, images, and movies. Images themselves can be selected to link to sounds or documents. Hypermedia simply combines hypertext and multimedia.

2) Humans associate pieces of information with other information and create complex knowledge structures. Hence, it is also said that the human memory is associative. For example, a person starts with an idea which reminds him/her of a related idea or a concept which in turn reminds him/her of another idea. The order in which a human associates an idea with another idea depends on the context under which the person wants information. When writing, an author converts his/her knowledge which exists as a complex knowledge structure into an external representation. Information can be represented only in a linear manner using physical media such as printed material and video tapes. Therefore, the author has to convert his knowledge into a linear representation using a linearisation process. The reading process can be viewed as a transformation of external information into an internal knowledge representation combined with integration into existing knowledge structures, basically a reverse operation of the writing process. For this, the reader breaks the information into

smaller pieces and rearranges these based on the readers information requirement. We rarely read a text book or a scientific paper from start to end. Hypermedia, using computer enabled links, allows us to partially imitate writing and reading processes as they take place inside our brain. We can create non linear information structures by associating pieces of information in different ways using links. Further, we can use a combination of media consisting of text, images, video, sound and animation for value addition in the representation of information. It is not necessary for an author to go through a linearisation process of his/her knowledge when writing. The reader can also access some of the information structures the author had when writing the information. This helps the readers create their own representation of knowledge and to gel it into existing knowledge structures.

3) Different types of links used in hypermedia are :

   Structural Links: The information contained within the hypermedia application is typically organised in some suitable fashion. This organisation is typically represented using structural links. We can group structural links together to create different types of application structures. If we look, for example, at a typical book, then this has both a linear structure i.e. from the beginning of the book linearly to the end of the book and usually a hierarchical structure in the form of the book that contains chapters, the chapters contain sections, the sections contains sub-sections etc. Typically in a hypermedia application we try to create and utilize appropriate structures.

   Associative Links: An associative link is a link which is completely independent of the specific structure of the information. We have links based on the meaning of different information components. The most common example which most people would be familiar with is cross-referencing within books for example - for more information on X refer to Y.

   Referential Links: A third type of link is a referential link. It is related to the associative link. Rather than representing an association between two related concepts, a referential link provides a link between an item of information and an explanation for that information.

Check Your Progress 2

1) Interactive: Users can use a variety of input devices to interact with the computer, such as a joystick, keyboard, touch screen, mouse, trackball, microphone, etc.

   Multi refers to the multiple file usages used in the multimedia product, such as sound, animation, graphics, video, and text.

   Media: Many media sources can be used as components in the multimedia product, such as a videodisk, CDROM, videotape, scanner, CD or other audio source, camcorder, digital camera, etc. Media may also refer to the storage medium used to store the interactive multimedia product, such as a videodisk or CDROM.

2) Multimedia is used in education and training fields as follows :
   - Computer simulations of things too dangerous, expensive, offensive, or time-sensitive to experience directly. Interactive tutorials that teach content by selecting appropriate sequencing of material based on the ongoing entry of student responses, while keeping track of student performance.
   - Electronic presentations.
   - Instruction or resources provided on the Internet (World Wide Web; 24 hours a day).
   - Exploratory hypertext software (i.e. encyclopedias, databases) used for independent exploration by learners to complete research for a paper, project,

or product development. They may use IMM resources to collect information on the topic or use multimedia components to create a product that blends visual, audio or textual information for effectively communicating a message.

Education courses, skills, and knowledge are sometimes taught out of context due to lack of application of real time examples. To overcome this, educators are using multimedia to bring into their classrooms real-world examples to provide a in-context framework important to learning. Multimedia and tools like the Internet give Faculty instant access to millions of resources.

Education training procedures fall into three general categories:

1) **Instructor Support products:** These are used by teachers in addition to text books, lectures and other activities within a class room environment.
2) **Standalone or Self paced products:** These are also called Computer based training and are designed for students to use in place of a teacher.
3) **Combination products:** As the name implies these fall between support and standalone products. These are used by students at the direction of instructors or to enhance classroom activities.

3) When a conference is conducted between two or more participants at different sites by using computer networks to transmit audio and video data, then it is known as video conferencing. A videoconference is a set of interactive telecommunication technologies which allow two or more locations to interact via two-way video and audio transmissions simultaneously. It has also been called visual collaboration and is a type of groupware.

   Digital compression of audio and video streams in real time is the core technology behind video conferencing. Codec is the hardware or software that performs compression. Compression rates of upto 1:500 can be achieved. The resulting digital stream of 1's and 0's is subdivided into labelled packets, which are then transmitted through a digital network usually ISDN or IP.

   The other components required for a VTC system include:

   Video input: video camera or webcam

   Video output: computer monitor or television

   Audio input: microphones

   Audio output: usually loudspeakers associated with the display device or telephone

   Data transfer: analog or digital telephone network, LAN or Internet.

Check Your Progress 3

1) Technology that makes computer capable of displaying and manipulating pictures.
2) Bitmap graphic images and vector graphic images.
3) The graphic images are quite useful on web but because of limit to the bandwidth of network it is necessary to compress and reduce the size of any image file, thus compression makes faster data access.
4) Since, JPEG file formats has incredible compression ratio of 1:100, so we can have better image in less size. Thus, JPEG file format is ideal for faster downloads.

Check Your Progress 4

1) **An analog recording** is one where the original sound signal is modulated onto another physical signal carried on some media or the groove of a gramophone disc or the magnetic field of a magnetic tape. A physical quantity in the medium (e.g., the intensity of the magnetic field) is directly related to the physical properties of the sound (e.g., the amplitude, phase and possibly direction of the sound wave.)

   A digital recording, on the other hand is produced by first encoding the physical properties of the original sound as digital information which can then, be decoded for reproduction. While it is subject to noise and imperfections in capturing the original sound, as long as the individual bits can be recovered, the nature of the physical medium is of minimum consequence in the recovery of the encoded information

2) There are three major groups of audio file formats:

   - common formats, such as WAV, AIFF and AU.
   - formats with lossless compression, such as FLAC, Monkey's Audio (filename extension APE), WavPack, Shorten, TTA, Apple Lossless and lossless Windows Media Audio (WMA).
   - formats with lossy compression, such as MP3, Vorbis, lossy Windows Media Audio (WMA) and AAC

3) DV Encoder Types
   DV Encoder Type 1
   DV Encoder Type 2
   Other Video File Formats
      AVI CODEC formats
      MPEG-1
      MPEG-2
      Quick time
      Real Video

4) The high bit rates that result from the various types of digital video make their transmission through their intended channels very difficult. Even entertainment videos with modest frame rates and dimensions would require bandwidth and storage space far in excess of that available on the CD-ROM. Thus, delivering consumer quality video on compact disc would be impossible.

   Similarly, the data transfer rate required by a video telephony system is far greater than the bandwidth available over the plain old telephone system (POTS). Even if high bandwidth technology (e.g. fiber-optic cable) was in place, the per-byte-cost of transmission would have to be very low before it would be feasible to use for the staggering amounts of data required by HDTV.

   Lastly, even if the storage and transportation problems of digital video were overcome, the processing power needed to manage such volumes of data would make the receiver hardware very

   This reduction of bandwidth has been made possible by advances in compression technology.

5) Progressive scan is used for most CRT computer monitors. (Other CRT-type displays, such as televisions, typically use interlacing.) It is also becoming increasingly common in high-end television equipment.

   With progressive scan, an image is captured, transmitted and displayed in a path similar to the text on a page: line by line, from top to bottom.

The interlaced scan pattern in a CRT (cathode ray tube) display would complete such a scan too, but only for every second line and then the next set of video scan lines would be drawn within the gaps between the lines of the previous scan.

Such scan of every second line is called a field.

The afterglow of the phosphor of CRT tubes, in combination with the persistence of vision results in two fields being perceived as a continuous image which allows the viewing of full horizontal detail but with half the bandwidth which would be required for a full progressive scan while maintaining the necessary CRT refresh rate to prevent flicker

Check Your Progress 5

1) Various types of basic tools for creating and editing multimedia elements are :

- Painting and Drawing tools
- Image Editing Tools
- OCR software
- 3-D Modeling and Animation tools
- Sound Editing Programs
- Animation, Video and Digital Movies.

2) Selection criteria for image editing applications are :

- Conversion of major image data types and industry standard file formats.
- Direct input from scanners etc.
- Employment of virtual memory scheme.
- Multiple window scheme.
- Image and balance control for brightness, contrast etc.
- Masking undo and restore features.
- Multiple video, Anti-aliasing, sharpening and smoothing controls.
- Color mapping controls.
- Geometric transformations.
- All colour palettes.
- Support for third party special effects plug ins.
- Ability to design in layers that can be combined, hidden and recorded.

3) Authoring tools usually refers to computer software that helps multimedia developers create products. Authoring tools are different from computer programming languages in that they are supposed to reduce the amount of programming expertise required in order to be productive. Some authoring tools use visual symbols and icons in flowcharts to make programming easier. Others use a slide show environment.

4) Authoring tools are grouped based on metaphor used for sequencing or organising multimedia elements and events

- Card or Page Based Tools
- Icon Based or Event Driven Tools
- Time Based and Presentation Tools
- Object Oriented Tools

2.9 FURTHER READINGS

Books

a) Multimedia Technology and Applications by David Hillman
b) Multimedia – Making it work by Tay Vaughan
c) Multimedia Systems Design by Prabhat K. Andleigh and Kiran Thakrar

Websites

a) www.en.wikipedia.org
b) www.computer.org
c) www.ieeecomputersociety.org
d) www.webopedia.com
e) www.fcit.usf.edu
f) www.why-not.com